



Apostila Linux

Apostila de introdução ao GNU/Linux

Objetivo do curso:

Esta apostila tem como objetivo ser uma ensinar aos estudantes e entusiastas o sistema operacional GNU/Linux informações para a operação e manutenção de um ambiente de trabalho utilizando o sistema operacional GNU/Linux (e outros tipos de sistemas operacionais baseados em Unix). Desta forma podemos preparar profissionais competentes para o mercado de trabalho.

Nota de Copyright:

Copyright © 2005 – Bruno César Brito Sant'Anna

Todo conteúdo aqui contido pode ser livremente copiado, distribuído e modificado mediante os termos da GNU Free Documentation License versão 1.1 ou posterior, publicado pela Free Software Foundation.

Fonte de pesquisa:

Grande parte desta apostila foi baseada no Guia FOCALinux© (<http://focalinux.cipsga.org.br/>) do autor Gleydson Mazioli da Silva. Também consultei o e-book Entendendo e dominando o Linux do autor Carlos E. Morimoto disponível no site <http://www.guiadohardware.net/ebooks/linux/index.html>. Muita coisa aqui escrita pode ser encontrada na internet utilizando ferramentas de busca. E-books da IBM Developer Works, Escritos por Daniel Robbins, Chris Houser e Aron Griffis disponíveis no site <http://www-106.ibm.com/developerworks/edu/l-dw-linuxlpi2-i.html>. Uma outra fonte de pesquisa utilizada nesta apostila e que pode ser de grande ajuda é o TLDP (The Linux Documentation Project <http://www.tldp.org/>)

Contato:

O contato pode ser feito por e-mail: brunocesar@ajato.com.br ou pelo site <http://gusl.usjt.objectis.net>

No site visite o fórum e deixe seu comentário, esta apostila esta em constante desenvolvimento todas as dicas são bem vindas.

Sobre o autor.

Bruno César Brito Sant'Anna é um universitário cursando Ciência da computação na Universidade São Judas Tadeu (USJT), atua como profissional de informática (administrador de rede, programador e técnico de suporte) nos últimos quatro anos. Tem alguns artigos publicados em sites de Linux:

- Utilizando certificados e-CNPJ e e-CPF no Linux:
<http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2466>
- Rede wireless: autenticação em uma rede WPA:
<http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2100>
- Monitorando o servidor Jabber 2 com o Bandersnatch:
<http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2335>
- Servidor Samba com antivírus Clamav:
<http://www.dicas-l.unicamp.br/dicas-l/20050203.php>

Recomendações:

Para um bom aproveitamento do conteúdo desta apostila sugiro que siga as seguintes recomendações:

- Pratique imediatamente o que aprendeu, desta forma o conhecimento recém adquirido será fixado com mais facilidade.
- É preciso interesse em aprender, se tiver vontade de aprender algo, você terá menos dificuldade do que algo que não gosta e está se obrigando.
- Decorar não adianta, pode até atrapalhar seus estudos, algumas vezes nos vemos forçados a decorar alguns comandos mas o objetivo é entender qual é a função deste comando.



Índice:

Prefacio

| | |
|--|----|
| Apresentação | 09 |
| Como será a sua aula | 09 |
| Requisitos necessários | 10 |
| Mercado de trabalho para administradores Linux | 10 |

Capítulo 1 – Conhecendo o GNU/Linux

| | |
|----------------------------|----|
| 1.1.O que é o GNU/Linux | 12 |
| 1.2.Características | 12 |
| 1.3.Distribuições de Linux | 13 |
| 1.4.Conceitos utilizados | 15 |
| 1.4.1.Kernel | 15 |
| 1.4.2.Módulos do Kernel | 15 |
| 1.4.3.Shell | 15 |
| 1.4.4.Arquivos | 16 |
| 1.4.5.Diretórios | 17 |
| 1.4.6.Login / Logout | 17 |
| 1.5.Exercícios | 17 |

Capítulo 2 – Introdução ao bash

| | |
|--------------------------------|----|
| 6.1.Bash | 19 |
| 6.2.Comando <code>cd</code> | 19 |
| 6.3.Paths | 20 |
| 2.3.1.Comando <code>pwd</code> | 20 |
| 2.3.2.Path absoluto | 20 |
| 2.3.3.Path relativo | 20 |
| 6.4.Atalhos para paths | 21 |
| 2.4.1.Atalho “.” | 21 |
| 2.4.2.Atalho “..” | 21 |
| 2.4.3.Atalho “~” | 21 |
| 2.4.4.Atalho “-” | 22 |
| 6.5.Exercícios | 22 |

Capítulo 3 – Comandos básicos.

| | |
|---|----|
| 3.1.Utilização de comandos no GNU/Linux e páginas do manual | 24 |
| 3.2.Comandos para manuseio de arquivos e diretórios | 25 |
| 3.2.1.Comando <code>touch</code> | 25 |
| 3.2.2.Comando <code>mkdir</code> | 25 |
| 3.2.3.Comando <code>mv</code> | 26 |

| | |
|----------------------------------|----|
| 3.2.4.Comando <code>cp</code> | 26 |
| 3.2.5.Comando <code>rm</code> | 26 |
| 3.2.6.Comando <code>rmdir</code> | 26 |
| 3.3.Comando <code>ls</code> | 27 |
| 3.4.Exercícios | 29 |

Capítulo 4 – Wildcards, links e redirecionamentos

| | |
|------------------------------|----|
| 4.1.Wildcards | 31 |
| 4.1.1.Sintaxe “ * ” | 31 |
| 4.1.2.Sintaxe “ ? ” | 31 |
| 4.1.3.Sintaxe “ [] ” | 32 |
| 4.1.4.Sintaxe “ [!] ” | 32 |
| 4.1.5.Formatção de wildcards | 32 |
| 4.2.Links | 33 |
| 4.2.1.Soft Links | 33 |
| 4.2.2.Hard Links | 34 |
| 4.3.Redirecionamento | 35 |
| 4.3.1.Pipe | 35 |
| 4.3.2.> | 36 |
| 4.3.3.>> | 36 |
| 4.3.4.<< | 37 |
| 4.4.Exercícios | 37 |

Capítulo 5 – Hierarquia do sistema de arquivos. Localizando arquivos e diretórios.

| | |
|---------------------------------------|----|
| 5.1.Hierarquia do sistema de arquivos | 40 |
| 5.2.Localizando arquivos | 41 |
| 5.2.1.O \$PATH | 41 |
| 5.2.2.Comando <code>whereis</code> | 41 |
| 5.2.3.Comando <code>locate</code> | 41 |
| 5.2.4.Comando <code>find</code> | 42 |
| 5.3.Exercícios | 43 |

Capítulo 6 – Expressões Regulares. Controle de Processos.

| | |
|------------------------------|----|
| 6.1.Expressões regulares | 45 |
| 6.1.1.Texto simples | 45 |
| 6.1.2.Sintaxe “ . ” | 45 |
| 6.1.3.Sintaxe “ [] ” | 45 |
| 6.1.4.Sintaxe “ [^] ” | 46 |
| 6.1.5.Sintaxe “ * ” | 46 |
| 6.1.6.Sintaxe “ ^ ” e “ \$ ” | 46 |

| | |
|--|----|
| 6.2. Controle de processos | 47 |
| 6.2.1. Iniciando um aplicativo | 47 |
| 6.2.2. Parando um aplicativo | 47 |
| 6.2.3. Listando processos | 47 |
| 6.2.4. Foreground e background | 48 |
| 6.2.5. Finalizando um aplicativo | 49 |
| 6.2.6. Iniciando um aplicativo em background | 49 |
| 6.2.7. Múltiplos processos em background | 50 |
| 6.2.8. Prioridades de processos | 50 |
| 6.3. Exercícios | 51 |

Capítulo 7 – Usuários, grupos e permissões de acesso

| | |
|--|----|
| 7.1. Usuários | 53 |
| 7.1.1. Usuário root | 53 |
| 7.1.2. Usuário normal | 53 |
| 7.1.3. Criando um usuário | 54 |
| 7.1.4. Modificando um usuário | 54 |
| 7.1.5. Modificando o password | 55 |
| 7.1.6. Excluindo um usuário | 55 |
| 7.2. Grupos | 55 |
| 7.2.1. Criando um grupo | 55 |
| 7.2.2. Excluindo um grupo | 56 |
| 7.3. Permissões de acesso | 56 |
| 7.3.1. Verificação de conta | 56 |
| 7.3.2. Verificação de permissões com <code>ls -l</code> | 57 |
| 7.3.3. Definindo posse de arquivos e diretórios por usuário com <code>chown</code> | 58 |
| 7.3.4. Definindo posse de arquivos e diretórios por grupo com <code>chgrp</code> | 58 |
| 7.3.5. O comando <code>chmod</code> | 59 |
| 7.3.6. Definindo permissões de acesso por triplets | 59 |
| 7.3.7. Definindo permissões de acesso pelo modo numérico | 60 |
| 7.4. Permissões especiais | 60 |
| 7.4.1. Umask | 60 |
| 7.4.2. Suid | 61 |
| 7.4.3. Sgid | 62 |
| 7.4.4. Sticky | 62 |
| 7.5. Exercícios | 62 |

Capítulo 8 – Instalando programas e trabalhando pacotes binários

| | |
|--------------------------------------|----|
| 8.1. Instalando novos programas | 64 |
| 8.2. Compilando pelo código-fonte | 64 |
| 8.2.1. Obtendo programas | 64 |
| 8.2.2. Descompactando/Desempacotando | 65 |

| | |
|---|----|
| 8.2.3.Arquivos README/INSTALL | 65 |
| 8.2.4.Script <code>configure</code> | 65 |
| 8.2.5.Compilando com <code>make</code> | 66 |
| 8.2.6.Instalando com <code>make install</code> | 66 |
| 8.2.7.Resumo da instalação por código fonte | 67 |
| 8.2.8.Resolvendo problemas | 67 |
| 8.2.9.Desinstalando com <code>make uninstall</code> | 67 |
| 8.3.Trabalhando com pacotes binários | 67 |
| 8.4.Empacotamento RPM | 68 |
| 8.4.1.Obtendo pacotes RPM | 69 |
| 8.4.2.Instalando pacotes RPM | 69 |
| 8.4.3.Resolvendo dependências | 69 |
| 8.4.4.Atualizando pacotes RPM | 70 |
| 8.4.5.Forçando a instalação de pacotes | 71 |
| 8.4.6.Gerenciamento de pacotes RPM | 71 |
| 8.4.7.Desinstalando pacotes RPM | 72 |
| 8.5.Empacotamento Debian | 73 |
| 8.5.1.Ferramenta <code>apt</code> | 73 |
| 8.5.2.O arquivo <code>/etc/apt/sources.list</code> | 75 |
| 8.5.3.Atualizando a lista de pacotes disponíveis via <code>apt</code> | 77 |
| 8.5.4.Instalando pacotes Debian via <code>apt</code> | 77 |
| 8.5.5.Atualizando pacotes Debian via <code>apt</code> | 77 |
| 8.5.6.Removendo pacotes via <code>apt</code> | 78 |
| 8.5.7.Outras formas de obter pacotes Debian | 78 |
| 8.5.8.Instalando / atualizando pacotes Debian | 78 |
| 8.5.9.Resolvendo dependências | 79 |
| 8.5.10.Forçando a instalação de pacotes Debian | 80 |
| 8.5.11.Desinstalando pacotes Debian | 80 |
| 8.6.Exercícios | 80 |

Capítulo 9 – Interface Gráfica: Servidor X

| | |
|--|----|
| 9.1.O que é o servidor X? | 82 |
| 9.1.1.Configurando o servidor X | 82 |
| 9.1.2.Iniciando e finalizando o servidor X | 83 |
| 9.2.Window Managers | 83 |
| 9.3.KDE | 83 |
| 9.3.1.Configurando o KDE | 84 |
| 9.3.2.Otimizando o KDE | 84 |
| 9.4.GNOME | 85 |
| 9.4.1.Configurando o GNOME | 85 |

| | |
|----------------------------|----|
| 9.4.2.Otimizando o GNOME | 86 |
| 9.5.Outros Window Managers | 86 |

Capítulo 10 – Configurando o Sistema

| | |
|--|----|
| 10.1.Introdução aos módulos do kernel | 88 |
| 10.2.Configurando placas de rede | 88 |
| 10.3.Configurando placas de rede Wi-Fi | 90 |
| 10.4.Configurando placas de som | 90 |
| 10.5.Configurando modem | 92 |
| 10.6.Configurando impressoras | 92 |
| 10.7.Configurando scanners | 94 |
| 10.8.Configurando multi-funcionais | 94 |
| 10.9.Configurando gravadores de CD/DVD | 95 |
| 10.10.Configurando câmeras digitais | 96 |

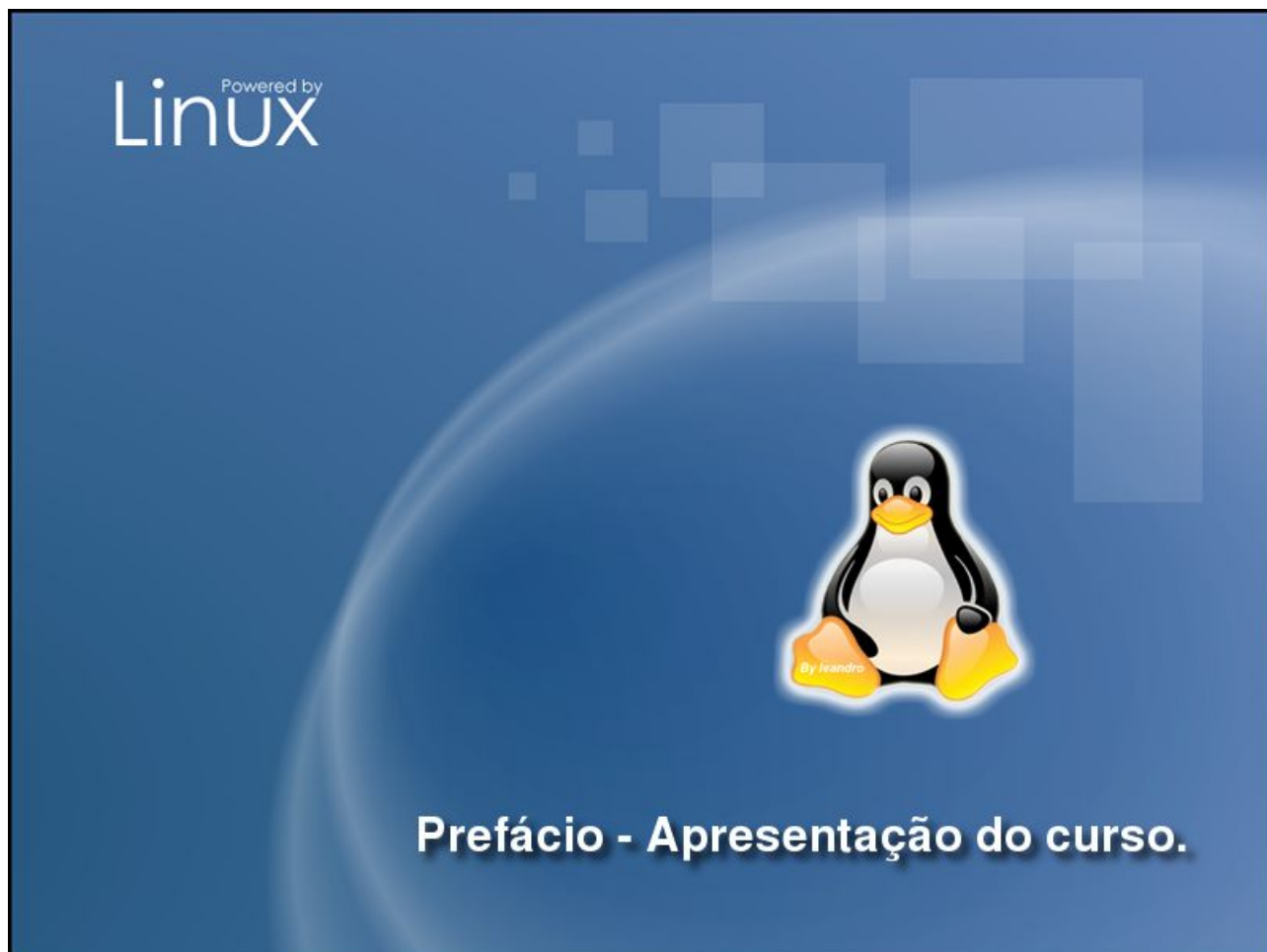
Capítulo 11 – Acessando a rede

| | |
|---|-----|
| 11.1.Configurando a rede | 98 |
| 11.2.Configurando via DHCP | 98 |
| 11.3.Acessando a Internet | 98 |
| 11.3.1.Discando via modem | 99 |
| 11.3.2.Acessando via banda-larga | 99 |
| 11.4.Acessando uma rede Windows | 100 |
| 11.4.1.Apresentando o Samba | 100 |
| 11.4.2.Configurando o Samba via SWAT | 100 |
| 11.4.3.Montando uma unidade de rede | 102 |
| 11.4.4.Compartilhando diretórios | 102 |
| 11.4.5.Compartilhando impressoras | 103 |
| 11.5.Acessando uma rede Linux | 103 |
| 11.5.1.Montando uma unidade de rede | 103 |
| 11.5.2.Compartilhando uma unidade de rede | 103 |

Apêndice A – Instalando o GNU/Linux (Distribuição Fedora Core 2)

| | |
|--|-----|
| A.1.Considerações iniciais | 106 |
| A.1.1.Onde conseguir o Linux? | 106 |
| A.1.2.Outros meios de instalação | 106 |
| A.2.Bootando o Linux | 107 |
| A.3.O Instalador | 107 |
| A.3.1.Definindo o perfil da instalação | 110 |
| A.3.2.Particionando o disco rígido | 111 |

| | |
|---|-----|
| A.3.3. Configurando o Bootloader (GRUB) | 114 |
| A.3.4. Configurando a rede | 114 |
| A.3.5. Idioma Padrão / Zona de Horário | 115 |
| A.3.6. Senha de root | 116 |
| A.3.7. Finalizando | 116 |



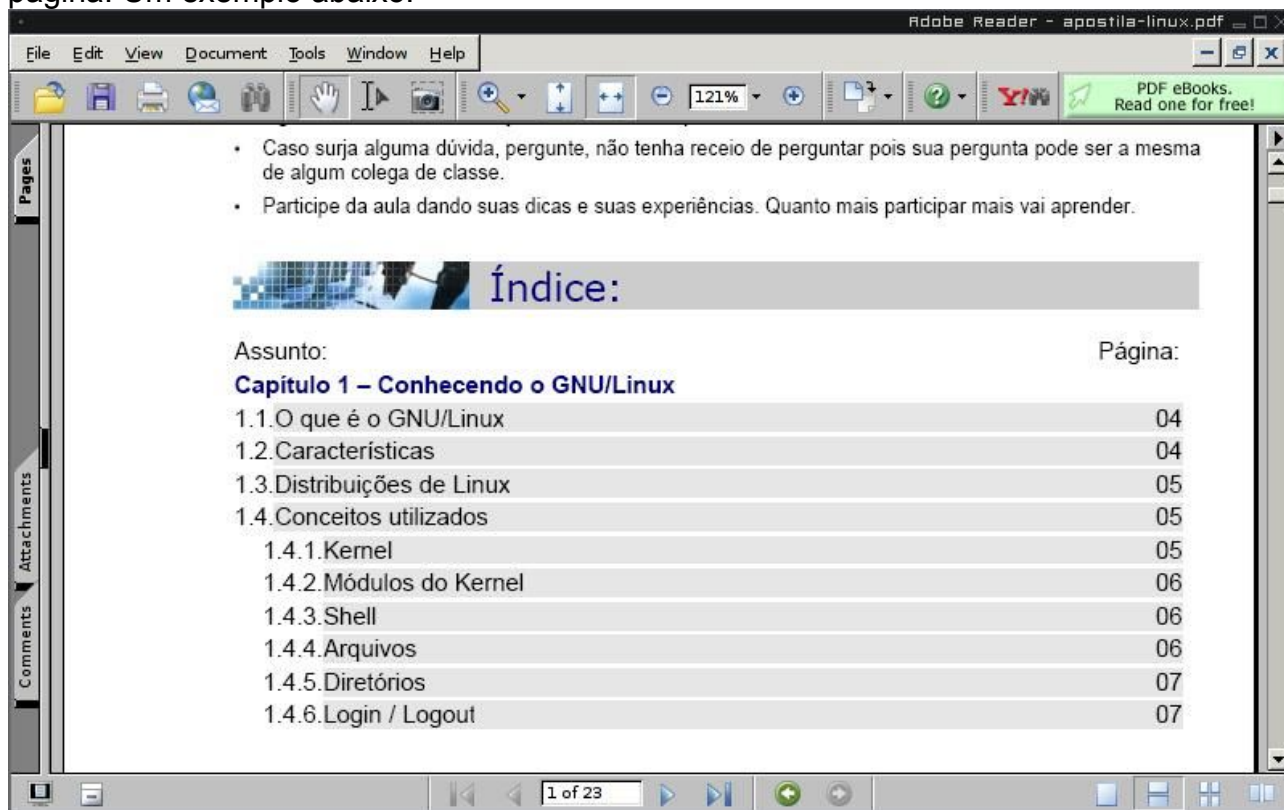
Aula inicial, serão apresentados os materiais do curso, a metodologia utilizada em classe e o professor. Também serão apresentados os requisitos necessários para o curso e uma perspectiva do mercado de trabalho.



Apresentação

Esta apostila foi desenvolvida especialmente para o curso de sistemas operacionais do IDEPAC, com os principais comandos e rotinas utilizadas no Linux, cobrindo grande parte das dúvidas e tarefas que um administrador de sistemas deve desempenhar em uma empresa. Serão cobertas desde a introdução ao Linux até rotinas avançadas de administração. Em todos os capítulos colocarei links para sites da internet onde podem ser encontradas mais informações sobre o tópico abordado.

Para melhorar o entendimento do curso, dividi em capítulos os temas do curso, todos podem ser acompanhados pelo índice que esta acima listado. Você provavelmente estará lendo esta apostila por um programa chamado Adobe Acrobat Reader, o que torna a navegação mais rápida, basta ir ao índice, verificar a página que contém o conteúdo da aula, depois disso vamos clicar na caixa branca abaixo do programa e especificamos a página. Um exemplo abaixo:



Exemplo de utilização do Acrobat na leitura da apostila.



Como será a sua aula

O computador que você vai utilizar já tem GNU/Linux instalado, a apostila será acompanhada pelo próprio computador, na lousa em todo início de aula estará escrito o tema e a página que se encontra o assunto do dia, cabendo-lhe a tarefa de ao iniciar a aula abrir a apostila e já ir para a página especificada na lousa. A navegação pela internet ficará desligada, podendo no entanto ser requisitada pelo professor quando for utilizada. Para um melhor aprendizado, recomendo que todas as aulas crie um arquivo simples de texto e escreva com suas palavras o que foi discutido, o que foi aprendido e salve em seu próprio disquete.

Ao final da aula haverá uma lista de exercícios para fixação de conteúdo

aprendido, com alguns comandos úteis.



Requisitos necessários

Por ser um sistema operacional avançado, o Linux requer alguns conhecimentos que foram abordados nos módulos anteriores do curso do IDEPAC e/ou conhecimentos equivalentes:

- Conhecimento básico em internet.
- Conhecimento básico em redes de computadores.
- Conhecimento com editores de texto.
- Conhecimento básico em Hardware.



Mercado de trabalho

O Linux é um sistema operacional com o foco corporativo, ou seja sua utilização é ampla em empresas, pois tem três fatores decisivos que tornam viável sua implementação; segurança, estabilidade e baixo custo. O mercado de trabalho para nós administradores de sistemas com conhecimento Linux é bem promissor, grande parte das empresas já estão em processo de migração. Um outro fator positivo para estudantes e profissionais de Linux, o governo brasileiro é favorável à implementação de software livre em repartições públicas e instituições governamentais em geral.

Por outro lado, a utilização em computadores pessoais, também chamados de desktops é baixa se comparada com outros sistemas operacionais, sendo utilizada em casos distintos. O Linux ainda está em processo de crescimento, nos próximos anos a tendência é de que se estabilize tanto no mercado corporativo quanto no mercado de computadores pessoais.



Neste capítulo será apresentado ao aluno o Sistema Gnu/Linux, suas características e seus principais conceitos. Ao final deste capítulo você será capaz de:

- Reconhecer um sistema GNU/Linux.
- Observar as principais características deste sistema operacional.
- Escolher, localizar e fazer o download de um sistema Linux distribuído pela internet por meio de *Distribuições Linux*.
- Entender os termos e conceitos do sistema.



1.1. O que é o GNU/Linux

O GNU/Linux, ou simplesmente Linux é um *sistema operacional* criado em 1991 por *Linus Torvalds* na universidade de Helsinki – Finlândia. É um sistema operacional de código aberto distribuído gratuitamente pela Internet. Você não precisa pagar nada para utilizar, não é crime fazer cópias e instalar em diversos computadores. Diante disto a comunidade Linux cresce de uma maneira impressionante, muitos desenvolvedores e usuários contribuem com o Linux fazendo otimizações e melhorias diversas. Um bom exemplo são as traduções feitas para o português do Brasil.



1.2. Características

A lista abaixo foi copiada do Guia FocaLinux.

- É livre e desenvolvido voluntariamente por programadores experientes, hackers, e contribuidores espalhados ao redor do mundo que tem como objetivo a contribuição para a melhoria e crescimento deste sistema operacional.
- Convive sem nenhum tipo de conflito com outros sistemas operacionais (com o DOS, Windows, OS/2) no mesmo computador.
- Multitarefa real .
- Multiusuário .
- Suporte a nomes extensos de arquivos e diretórios (255 caracteres) .
- Conectividade com outros tipos de plataformas como *Apple, Sun, Macintosh, Sparc, Alpha, PowerPc, ARM, Unix, Windows, DOS, etc.*
- Proteção entre processos executados na memória RAM .
- Suporte a mais de 63 terminais virtuais (consoles) .
- Modularização - O GNU/Linux somente carrega para a memória o que é usado durante o processamento, liberando totalmente a memória assim que o programa/dispositivo é finalizado .
- Devido a modularização, os drivers dos periféricos e recursos do sistema podem ser carregados e removidos completamente da memória RAM a qualquer momento.
- Não há a necessidade de se reiniciar o sistema após a modificar a configuração de qualquer periférico ou parâmetros de rede. Somente é necessário reiniciar o sistema no caso de uma instalação interna de um novo periférico, falha em algum hardware (queima do processador, placa mãe, etc.).
- Não precisa de um processador potente para funcionar. O sistema roda bem em computadores 386Sx 25 com 4MB de memória RAM (sem rodar o sistema gráfico X, que é recomendado 8MB de RAM).
- O crescimento e novas versões do sistema não provocam lentidão, pelo contrário, a cada nova versão os desenvolvedores procuram buscar maior compatibilidade, acrescentar recursos úteis e melhor desempenho do sistema .
- Não é requerida uma licença para seu uso. O GNU/Linux é licenciado de acordo com os termos da GPL.
- Acessa corretamente discos formatados pelo DOS, Windows, Novell, OS/2, NTFS, SunOS, Amiga, Atari, Mac, etc.
- Utiliza permissões de acesso a arquivos, diretórios e programas em execução na memória RAM.
- NÃO EXISTEM VÍRUS NO LINUX! Em 13 anos de existência, nunca foi registrado NENHUM tipo de infecções desde que respeitadas as recomendações padrão de política de segurança e uso de contas privilegiadas (como a de root, como veremos adiante).
- Rede TCP/IP mais rápida que no Windows e tem sua pilha constantemente melhorada. O GNU/Linux tem suporte nativo a redes TCP/IP e não depende de uma camada intermediária como o WinSock. Em acessos via modem a Internet, a velocidade de transmissão é 10% maior.
- Roda aplicações *DOS* através do DOSEMU
- Roda aplicações *Windows* através do WINE.
- Suporte a rede via rádio amador.
- Suporte a dispositivos Plug-and-Play.

- Suporte a dispositivos USB.
- Vários tipos de firewalls de alta qualidade e com grande poder de segurança de graça.
- Possui recursos para atender a mais de um endereço IP na mesma placa de rede, sendo muito útil para situações de manutenção em servidores de redes ou para a emulação de "mais computadores" virtualmente.
- O sistema de arquivos usado pelo GNU/Linux (Ext3) organiza os arquivos de forma inteligente evitando a fragmentação e fazendo-o um poderoso sistema para aplicações multi-usuárias exigentes e gravações intensivas.
- Permite a montagem de um servidor Web, E-mail, News, etc. com um baixo custo e alta performance. O melhor servidor Web do mercado, o Apache, é distribuído gratuitamente junto com o Linux. O mesmo acontece com o Sendmail.
- Por ser um sistema operacional de código aberto, você pode ver o que o código fonte (o que foi digitado pelo programador) faz e adapta-lo as suas necessidades ou de sua empresa. Esta característica é uma segurança a mais para empresas sérias e outros que não querem ter seus dados roubados.
- Suporte a diversos dispositivos e periféricos disponíveis no mercado, tanto os novos como obsoletos.





1.3. Distribuições de Linux





Por ser um sistema livre, surgiram grupos de pessoas, empresas e organizações que resolveram distribuir o Linux com pacotes de programas, instaladores próprios e algumas customizações, que diferenciam-se entre si. Este é o conceito principal de distribuição no Mundo Linux.

Existem diversas distribuições distintas, cada uma com suas características. Algumas, recomendadas para iniciantes, outras para usuários avançados. A escolha de uma distribuição é uma questão de gosto.

Neste curso utilizaremos a distribuição **Debian** porem aprenderemos conceitos das distribuições mais utilizadas no mercado.

Segue abaixo a relação das principais distribuições :

| | | |
|-----------------------------|--|---|
| Nome: | Debian |  |
| Site: | www.debian.org | |
| Característica: | O Debian é considerado pelos profissionais uma distribuição estável pois todos os programas inclusos são rigorosamente testados (testes que levam de dois a três anos). É desenvolvido por profissionais e não tem vínculos com nenhuma empresa. | |
| Nível de utilização: | Intermediário / Avançado. | |
| Nome: | Conectiva |  |
| Site: | www.conectiva.com.br | |
| Característica: | O foco da distribuição Conectiva é o público brasileiro, é de fácil utilização, tem suporte nacional (pago). É desenvolvida pela Empresa Conectiva S/A. esta possui diversos cursos de especialização. | |
| Nível de utilização: | Iniciante. | |

| | | |
|-----------------------------|---|---|
| Nome: | Red Hat |  |
| Site: | www.redhat.org | |
| Característica: | A distribuição norte americana Red Hat Linux é bem conhecida pelos profissionais pois além de ser antiga, é bem documentada de fácil utilização e pode ser usada tanto em Desktops quanto em Servidores. Deu origem ao projeto Fedora. | |
| Nível de utilização: | Iniciante / Intermediário. | |
| Nome: | Slackware |  |
| Site: | www.slackware.com | |
| Característica: | A distribuição Slackware já é um pouco mais difícil, a maioria das configurações dela são feitas editando manualmente arquivos de texto, isto é um pesadelo para iniciantes mas por outro lado torna-se uma distribuição extremamente customizável, proporcionando um contato direto com o Linux. Grande parte de sua documentação encontra-se em inglês. | |
| Nível de utilização: | Intermediário / Avançado. | |
| Nome: | SuSe |  |
| Site: | www.novell.com/linux/suse/ | |
| Característica: | A distribuição alemã SuSe é de fácil utilização, tem grande suporte à hardware. É uma distribuição ideal para quem está começando pois é muito amigável e tem um sistema de instalação de programas robusto. Pode ser utilizada tanto em Desktops quanto em Servidores. | |
| Nível de utilização: | Iniciante. | |
| Nome: | Kurumin |  |
| Site: | www.kurumin.org | |
| Característica: | A distribuição brasileira Kurumin é de fácil utilização e tem como foco o usuário iniciante, roda direto do CD-Rom e não é necessária a sua instalação. Eu recomendo pessoalmente que seja utilizada só em primeiros contatos com o Linux pois sua utilização fica limitada à sua interface gráfica. Usuários que já tem conhecimento no sistema Linux podem utilizar o console de comando mas suas dependências de pacotes e conflitos de hardware a tornam uma distribuição não indicada para utilização à longo prazo. | |
| Nível de utilização: | Iniciante. | |

Foram mencionadas acima somente as distribuições mais conhecidas e comentadas nos últimos tempos, uma boa fonte de pesquisa à demais distribuições é no site <http://www.linuxiso.org>, onde também serão encontradas seus CD's para download.



1.4. Conceitos utilizados

No mundo Linux, veremos uma série de termos estranhos a princípio mas que se tornarão comuns em nosso curso e posterior carreira. Termos o quais nunca ouvimos falar tais como o Kernel, shell e etc. mas que são tão importantes para o conhecimento do sistema. Segue abaixo uma explicação sobre os mesmos:

1.4.1. Kernel

Sendo bem claro, o Kernel é próprio Linux, ele é o coração do sistema, que controla todos os dispositivos do computador (como memória, placas de som, vídeo, discos rígidos, disquetes, sistemas de arquivos, redes e outros recursos disponíveis). O Kernel do linux tem o código aberto, desta forma, todos podem editar e compilar o Kernel conforme nossas necessidade, habilitando suporte a novos dispositivos.

Mais adiante no curso aprenderemos como modificar e recompilar o Kernel do Linux para habilitar e desabilitar determinadas características, inclusive a criação de módulos do kernel, explicados abaixo.

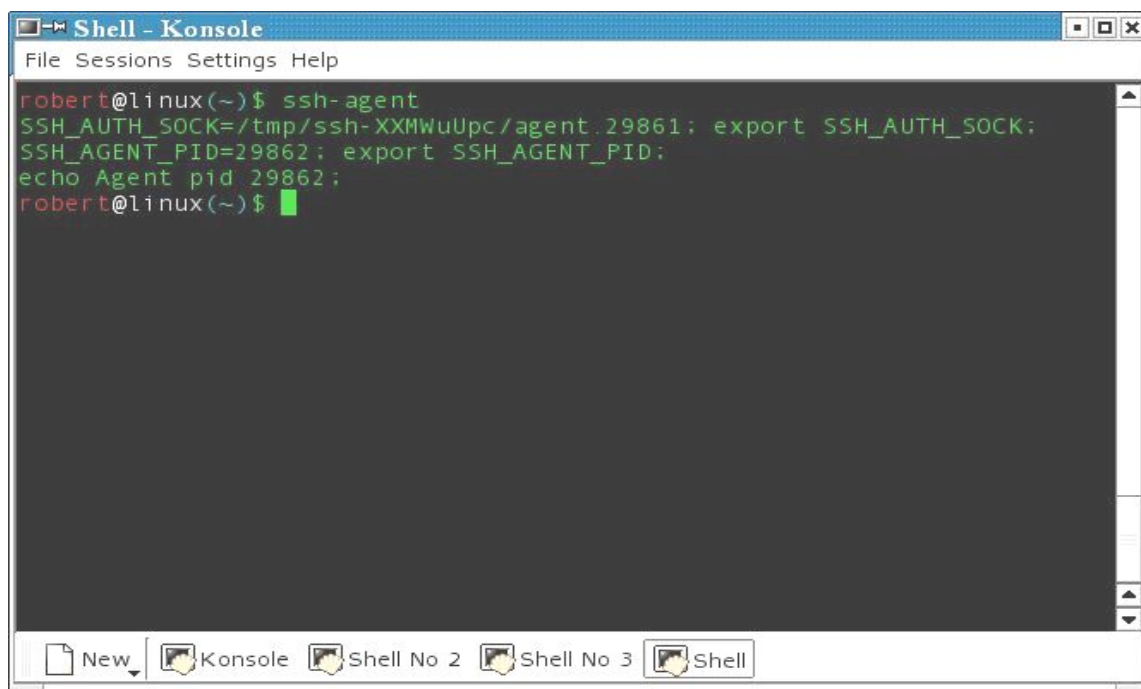
O Kernel do Linux é desenvolvido por um time de profissionais e pode ser acompanhado pelo seu site oficial: www.kernel.org.

1.4.2. Módulos do Kernel

Módulos são partes do Kernel que são carregadas na execução do sistema com o objetivo de implementar um novo recurso ou serviço ao sistema operacional. Podem ser carregados a qualquer hora e descarregados quando não forem mais necessários, isto é útil pois economiza recursos computacionais e não requer a inicialização do SO. Um exemplo prático; para se utilizar uma placa de rede precisamos de um driver certo? Nós indicamos com o comando `modprobe modulo_da_placa` e nossa placa vai ser habilitada no sistema, caso não precisemos mais utilizar a placa digitamos `rmmod modulo_da_placa` e a placa será desabilitada, trataremos de módulos detalhadamente adiante.

1.4.3. Shell

Shell é um interpretador de comandos, ou seja, é ele quem traduz uma ordem dada pelo usuário via teclado ao Kernel, existem diversos shell no linux sendo que o padrão é o Bash. Através do shell controlamos o sistema operacional, a utilização do shell pode parecer primitiva, um usuário de Windows raramente precisa entrar no shell para realizar suas operações, mas no linux as coisas mudam de figura, todos os programas rodam a partir de um shell inclusive a interface gráfica. Mas então toda vez que formos iniciar um programa precisamos iniciar um shell antes? A resposta é não. Podemos abrir programas quando estivermos na interface gráfica com cliques em cima dos ícones. Comandos executados no shell podem executar tarefas de uma maneira muito mais rápida do que na parte gráfica. Aprenderemos diversos comandos do shell adiante.



```
Shell - Konsole
File Sessions Settings Help

robert@linux(~)$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-XXMWuUpC/agent.29861; export SSH_AUTH_SOCK;
SSH_AGENT_PID=29862; export SSH_AGENT_PID;
echo Agent pid 29862;
robert@linux(~)$
```

No caso do exemplo acima, o Terminal é um programa chamado Konsole.

No GNU/Linux, em modo texto, você pode acessar outros terminais virtuais segurando a tecla ALT e pressionando F1 a F6. Cada tecla de função corresponde a um número de terminal do 1 ao 6 (o sétimo é usado por padrão pelo ambiente gráfico X). O GNU/Linux possui mais de 63 terminais virtuais, mas apenas 6 estão disponíveis inicialmente por motivos de economia de memória RAM (cada terminal virtual ocupa aproximadamente 350 Kb de memória RAM).

1.4.4. Arquivos

Um arquivo pode conter um texto feito por nós, uma música, programa, planilha, etc. Cada arquivo deve ser identificado por um nome, assim ele pode ser encontrado facilmente quando desejar usa-lo. Se estiver fazendo um trabalho de história, nada melhor que salva-lo com o nome *historia*. Um arquivo pode ser binário ou texto.

texto

Seu conteúdo é compreendido pelas pessoas. Um arquivo texto pode ser uma carta, um script, um programa de computador escrito pelo programador, arquivo de configuração, etc.

binário

Seu conteúdo somente pode ser entendido por computadores. Contém caracteres incompreensíveis para pessoas normais. Um arquivo binário é gerado através de um arquivo de programa (formato texto) através de um processo chamado de **compilação**. Compilação é basicamente a conversão de um programa em linguagem humana para a linguagem de máquina.

O GNU/Linux é *Case Sensitive* ou seja, ele diferencia letras *maiúsculas* e *minúsculas* nos arquivos. O arquivo *historia* é completamente diferente de *Historia*. Esta regra também é válido para os *comandos* e *diretórios*. Prefira, sempre que possível, usar letras minúsculas para identificar seus arquivos, pois quase todos os comandos do sistema estão em *minúsculas*.

Um arquivo oculto no GNU/Linux é identificado por um "." no início do nome (por exemplo, .bashrc). Arquivos ocultos não aparecem em listagens normais de diretórios, deve ser usado o comando `ls -a` para também listar arquivos ocultos.

1.4.5. Diretórios

Diretório é o local utilizado para armazenar conjuntos arquivos para melhor organização e localização. O diretório, como o arquivo, também é "Case Sensitive" (diretório /teste é completamente diferente do diretório /Teste). Não podem existir dois arquivos com o mesmo nome em um diretório, ou um sub-diretório com um mesmo nome de um arquivo em um mesmo diretório.

Um diretório nos sistemas Linux/UNIX são especificados por uma "/" e não uma "\" como é feito no DOS.

Diretórios também podem ser ocultos utilizando o "." antes do nome.

1.4.6. Login / Logout

Login é a primeira coisa que aparece quando você inicia o Linux, ele pede usuário e senha assim como sites de webmail, desta forma o sistema tem um controle de dos usuários que acessam a maquina, definindo as suas permissões. Logout é a saída do sistema quando são digitados `logout`, `exit`, `CTRL+D`.



Exercícios de fixação

Para fixar o conteúdo aprendido execute os exercícios abaixo:

1. Abra um shell, e utilize os comandos:
 - a. `uname -a` para exibir as informações sobre seu sistema (Respecitamente: Sistema operacional, nome do host, versão do kernel, data e arquitetura.)
 - b. `lsmod` para exibir os módulos do kernel carregados atualmente.
 - c. `echo $SHELL` para exibir o shell que esta sendo utilizando no momento.
 - d. `mkdir primeirodiretorio` para criar um diretório chamado **primeirodiretorio**.
 - e. `touch primeiroarquivo` para criar um arquivo sem conteúdo chamado **primeiroarquivo**.
 - f. `ls` para listar os arquivos e os diretórios localizados no local atual (diretório atual).
2. Crie um arquivo oculto, depois de criado, demonstre que este arquivo existe utilizando o comando `ls -a`.
3. Efetue o logout do shell atual utilizando o comando `logout`.



No capítulo anterior vimos o shell, e vimos que o padrão no GNU/Linux é o bash, neste capítulo veremos mais funções do bash, comandos básicos de navegação paths e atalhos de paths. Ao final deste capítulo você será capaz de:

- Navegar corretamente por diretórios no GNU/Linux
- Distinguir paths absolutos e paths relativos
- Dominar os atalhos de navegação



2.1. Bash

Bash é o uma abreviação de “*bourne shell again*”, vimos no capítulo anterior que ele é o shell padrão do GNU/Linux. Apesar de sua função principal ser interpretar comandos dados e passa-los ao kernel, existem outros aspectos importantes que devemos abordar antes de prosseguir no curso. Quando estamos logados no bash a seguinte linha aparece no console:

```
bruno@Shinji:~$
```

Esta linha representa respectivamente; o usuário logado, o símbolo @ (arroba que em inglês significa at, at em português significa “em”) o hostname (nome da máquina), o diretório atual (veremos adiante neste capítulo que o símbolo ~ indica diretório home) e finalmente o símbolo \$.

Quando temos o símbolo \$ no final da linha, significa que estamos trabalhando com um usuário normal, ou seja, sem poderes de administrador. O outro símbolo que podemos encontrar no final da linha é o #, significa que estamos trabalhando como o usuário root, o usuário root tem poderes de administrador, explicaremos usuários nos capítulos posteriores.

Desta forma temos certeza de que tipo de conta estamos utilizando, geralmente quando temos um tutorial na internet explicando como instalar algum programa ou outra tarefa similar via console, os autores utilizam estes símbolos para determinar qual comando pode ser executado como usuário normal e qual deve ser executado como usuário root.

O bash também permite a criação de “programas” simples, chamados *shell scripts*, no Apêndice B veremos um tutorial de *shell scripts*.

Outra característica importante do bash é que ele pode armazenar variáveis de ambiente, o nome do usuário por exemplo fica armazenado na variável **\$USERNAME**, assim como diversas outras variáveis, que podem ser mudadas conforme a necessidade, uma relação completa das variáveis de um bash atual pode ser obtida com o comando *set*.



2.2. Comando *cd*

Este é tranquilamente o comando mais utilizado no ambiente Linux, serve somente para mudar de um diretório para outro, permitindo assim a navegação pelo sistema de arquivos:

```
bruno@Shinji:~$ cd /
```

Com o comando acima mudamos para o diretório “/” também conhecido como diretório root, que é o diretório que está no topo da árvore de diretórios.



2.3 Paths

Path em inglês significa “caminho”, path nada mais é do que o diretório ou caminho para determinado arquivo ou programa.

2.3.1. Comando `pwd`

O comando `pwd` retorna o path atual (ou diretório atual):

```
bruno@Shinji:/$ pwd
/
```

Como anteriormente mudamos para o diretório `/` com o comando `cd` o path atual torna-se o diretório `/`, vamos agora para um outro diretório e verificar o path:

```
bruno@Shinji:/$ cd /tmp
bruno@Shinji:/tmp$ pwd
/tmp
```

O diretório para o qual mudamos é um diretório absoluto.

2.3.2. Path absoluto

Paths absolutos são os caminhos completos para determinado diretório, sempre começam com o caractere “ `/` ”, o que significa que é o caminho desde o diretório root. Abaixo exemplos de outros paths absolutos:

```
/dev
/usr
/usr/bin
/usr/local/bin
```

2.3.3. Path relativo

Paths relativos são os caminhos para diretórios que estão dentro do path atual, nunca começam com o caractere “ `/` ”. Exemplificando; vamos supor que estejamos no diretório `/usr`:

```
bruno@Shinji:/tmp$ cd /usr
bruno@Shinji:/usr$ pwd
/usr
```

Para ir para o diretório `/usr/local/bin` usamos o path relativo:

```
bruno@Shinji:/usr$ cd local/bin
bruno@Shinji:/usr/local/bin$ pwd
/usr/local/bin
```

Omitimos os campos “/usr/” para ir ao diretório /usr/local/bin. Veja bem que o diretório “/local” esta dentro de “/usr”.



2.4. Atalhos para paths

Para facilitar a navegação foram inventados atalhos para paths mais utilizados.

2.4.1. Atalho “.”

E o atalho para o diretório atual, utilizado quando temos que executar algum arquivo que esta no diretório atual, podemos verificar sua existência com o comando `ls -a` :

```
bruno@Shinji:/usr/local/bin$ cd /tmp
bruno@Shinji:/tmp$ mkdir novodir
bruno@Shinji:/tmp$ cd novodir
bruno@Shinji:/tmp/novodir$ ls -a
.  ..
```

Suponhamos que tenha um programa que queremos executar e este esta no path atual, então executamos desta forma: “./programa”.

2.4.2. Atalho “..”

E o atalho para o diretório acima do atual:

```
bruno@Shinji:/tmp/novodir$ pwd
/tmp/novodir
bruno@Shinji:/tmp/novodir$ cd ..
bruno@Shinji:/tmp$ pwd
/tmp
```

2.4.3. Atalho “~”

E o atalho para o diretório home do usuário atual:

```
bruno@Shinji:/tmp$ cd ~
bruno@Shinji:~$ pwd
/home/bruno
```

Essa linha é familiar não? E a mesma linha de quando iniciamos um console :-)

Caso precisemos ir para o diretório home de um outro usuário procedemos da seguinte maneira:

```
bruno@Shinji:~$ cd ~italo
bruno@Shinji:/home/italo$ pwd
/home/italo
```

2.4.4. Atalho “-”

Com este atalho voltamos para o diretório que estávamos trabalhando anteriormente:

```
bruno@Shinji:/home/italo$ cd -
bruno@Shinji:~$ pwd
/home/bruno
```



Exercícios de fixação

Para fixar o conteúdo aprendido execute os exercícios abaixo:

1. Treine a navegação com o comando `cd`, mude para os diretórios `/usr`, `/tmp`, `/usr/local/`.
2. Analise se os paths abaixo são relativos ou absolutos:
 - a. `/usr/local/bin`
 - b. `local/bin`
 - c. `/home/bruno`
 - d. `/etc`
 - e. `..`
 - f. `/usr/local/./local/bin`
 - g. `~`
3. Treine a navegação para os atalhos `..`, `~` e `-`.
4. Vá para o diretório `/aulas/aula2` e execute o arquivo `script1.sh` com o atalho `..`.



No capítulo anterior vimos o shell com certa profundidade e os atalhos de navegação por diretórios relativos e absolutos. Neste capítulo iremos aprender mais alguns comandos do GNU/Linux e a utilização destes comandos. Ao final deste capítulo você será capaz de:

- Empregar corretamente as sintaxes de utilização de comandos no GNU/Linux
- Consultar páginas do manual sobre os comandos do GNU/Linux
- Manusear arquivos e diretórios
- Listar o conteúdo de diretórios
- Distinguir links, diretórios de arquivos comuns



3.1. Utilização de comandos no GNU/Linux e páginas do manual

Grande parte dos comandos do GNU/Linux contém uma ou mais opções (geralmente estas opções servem para formatar a saída do comando), desta forma a sua utilização depende de uma sintaxe, ou seja empregar o comando corretamente. A forma básica de um comando é esta:

```
$ comando -x --x_por_extenso <informação adicional>
```

Traduzindo a sintaxe acima:

| | |
|---------------------|--|
| \$ | Bash com usuário normal |
| comando | O comando que iremos executar |
| -x | Opção de execução do comando, é opcional para formatar a saída do comando, um comando <u>pode</u> ter mais de uma opção por vez |
| -- x_por_extenso | Uma variação de como se escreve a opção, para fácil memorização, geralmente um comando tem uma variação desta forma. por exemplo -l = --list Aparece com -- para evitar confusões. |
| < informação > | Informação necessária para a utilização do comando. por exemplo cd /diretorio no caso o "/diretorio" é esta informação. |

Para obtermos uma lista completa da utilização de um determinado comando do GNU/Linux consultamos as páginas do manual. O manual do GNU/Linux é um manual completo com documentação de comandos e programas do GNU/Linux. Utilização:

```
$ man comando
```

Por ser um sistema operacional gratuito, podem estar faltando manuais de alguns comandos ou os mesmos podem estar desatualizados. De qualquer forma o manual deve ser consultado primeiro em qualquer ocasião que precise de ajuda. A navegação é efetuada com as setas e quando acabar de ler o manual do comando digite "q" que fecha a pagina imediatamente. Abaixo a forma como as paginas de manual são formatadas:

| | |
|-------------|--|
| NAME | Nome do comando e descrição de e uma linha. |
| SYNOPSIS | Como utilizar o comando. |
| DESCRIPTION | Descrição aprofundada sobre a funcionalidade do comando. |

| | |
|----------|--|
| EXAMPLES | Exemplos e sugestões de como utilizar o comando. |
| SEE ALSO | Tópicos relacionados (geralmente outras paginas de manual) |

Estudaremos mais sobre paginas de manual em capítulos posteriores.



3.2 Comando para manuseio de arquivos e diretórios.

Entende-se por manuseio a criação, alteração, localização e exclusão de determinado arquivo e/ou diretório. Para meios didáticos, cobrirei a localização de arquivos no próximo capítulo, abaixo vou demonstrar comandos de criação, deslocamento e remoção de arquivos e diretórios.

3.2.1. Comando `touch`

O comando `touch` serve, na verdade, para atualizar o horário de modificação de determinado arquivo, se o mesmo não existir ele cria o arquivo sem conteúdo.

```
$ touch novoarquivo.txt
```

O comando acima criou o arquivo **novoarquivo.txt**.

3.2.2. Comando `mkdir`

O comando `mkdir` serve para criar um diretório vazio.

```
$ mkdir dir1
```

O diretório **dir1/** foi criado. Veja agora o exemplo abaixo:

```
$ mkdir dir2/dir3
mkdir: cannot create directory 'dir2/dir3': No such file or directory
```

Isto resulta em erro pois o comando `mkdir` só pode construir um diretório por vez, ou seja o diretório **dir3** não pode ser criado se o diretório **dir2** não existe, a maneira correta seria:

```
$ mkdir dir2
$ mkdir dir2/dir3
```

Ou então simplesmente:

```
$ mkdir -p dir2/dir3
```

A opção “-p” permite a criação de múltiplos diretórios de uma só vez.

3.2.3. Comando `mv`

O comando `mv` serve para mover arquivos e diretórios de um diretório para o outro, serve também para renomear arquivos e diretórios caso sejam “movidos” para o mesmo local que estão mas com nomes diferentes.

```
$ mv novoarquivo.txt dir1/
```

O comando acima moveu o arquivo **novoarquivo.txt** para o diretório **dir1/**.

O mesmo pode ocorrer com diretórios:

```
$ mv dir2/dir3/ dir1/
```

O comando acima moveu o diretório **dir3/** de dentro do diretório **dir2/** para o dentro do diretório **dir1/**.

3.2.4. Comando `cp`

O comando `cp` copia arquivos e diretórios.

```
$ cp dir1/novoarquivo.txt dir2/
```

O comando acima copiou o arquivo **novoarquivo.txt** que estava no diretório **dir1/** para o diretório **dir2/**, Agora ambos os diretórios **dir1** e **dir2** possuem o arquivo **novoarquivo.txt**. Veja agora o exemplo abaixo:

```
$ cp dir2 dir1  
cp: omitting directory `dir2/`
```

A segunda linha demonstra o erro resultante na tentativa de copia de um diretório para outro, para corrigir este erro utilize a opção `-r`.

```
$ cp -r dir2 dir1
```

3.2.5. Comando `rm`

O comando `rm` remove os arquivos especificados.

```
$ rm dir2/novoarquivo.txt
```

O arquivo **novoarquivo.txt** que estava no diretório **dir2/** foi removido.

3.2.5. Comando `rmdir`

O comando `rmdir` remove os diretórios especificados.

```
$ rmdir dir2/dir3
```

O diretório **dir3** que estava dentro do diretório **dir2/** foi removido, note que o diretório **dir2** continua intacto, vamos remove-lo também:

```
$ rmdir dir2
```

Pronto, o diretório **dir2** também foi removido, veja agora o exemplo abaixo:

```
$ rmdir dir1
rmdir: `dir1/`: Directory not empty
```

Deu erro! Explicação; o diretório **dir1** não estava vazio, ou seja, somente diretórios vazios podem ser removidos.

Nestes casos que temos um diretório com algo dentro imagine se tivessem milhares de arquivos dentro de um diretório, ou pior, milhares de diretórios dentro de diretórios dentro de mais diretórios dentro deste diretório que você está tentando apagar. Isto levaria dias

Para estes casos existe o comando **rm** com os parâmetros **-rf**. Que simplesmente apaga tudo o que tem no diretório e logo após apaga o diretório em si, este comando é muito perigoso e serve somente para casos extremos, pois pode inutilizar o sistema.

```
$ rm -rf dir1
```

O comando acima vai deletar o diretório **dir1** e todo seu conteúdo não importando o que haja nele, por isso deve ser utilizado com cuidado.



3.3. Comando **ls**

Este comando lista na tela o conteúdo dos diretórios:

```
$ cd /usr
$ ls
X11R6  bin  doc  games  include  info  lib  local  man  sbin
share  src
```

Especificando a opção **-a** podemos ver todos os arquivos, inclusive os ocultos:

```
$ ls -a
.  ..  X11R6  bin  doc  games  include  info  lib  local  man
sbin  share  src
```

Especificando a opção **-l** podemos ver todos os arquivos, inclusive os ocultos:

```
$ ls -l
drwxr-xr-x    6 root   root    4096 Jan 11 06:23 X11R6
drwxr-xr-x    2 root   root   49152 May  5 14:43 bin
drwxr-xr-x    2 root   root   12288 Apr 28 13:58 doc
drwxr-xr-x    2 root   root    4096 Apr  6 14:52 games
drwxr-xr-x   78 root   root    8192 May  3 17:01 include
lrwxrwxrwx    1 root   root      10 Jan 11 09:41 info -> share/info
drwxr-xr-x  160 root   root   53248 May  5 14:43 lib
drwxrwsr-x   15 root  staff    4096 Apr 18 10:14 local
drwxr-xr-x    4 d3v1l  users    4096 Jan 24 14:34 man
drwxr-xr-x    2 root   root    8192 May  5 11:55 sbin
drwxr-xr-x  239 root   root    8192 May  5 11:55 share
drwxrwsr-x    9 root   src      4096 Apr 11 09:03 src
```

Esta saída traz muitas informações, vamos utilizar como exemplo primeira linha da saída para explicar a sintaxe utilizada pelo comando `ls -l`.

| drwxr-xr-x 6 root root 4096 Jan 11 06:23 X11R6 | |
|--|--|
| drwxr-xr-x | “Triplet” que mostra o tipo de arquivo e a permissão de acesso ao arquivo. |
| 6 | Número de Links que o arquivo possui, links são atalhos. |
| root | Usuário dono do arquivo. |
| root | Grupo dono do arquivo. |
| 4096 | Tamanho do arquivo em bytes. |
| Jan 11 06:23 | Data de última modificação do arquivo. |
| X11R6 | Nome do arquivo. |

Agora vamos explicar esta linha como deve ser interpretada pelo usuário:

| drwxr-xr-x 6 root root 4096 Jan 11 06:23 X11R6 | |
|--|--|
| drwxr-xr-x | d significa diretório r significa read (leitura em inglês) w significa write (escrita em inglês) x significa execute (executar em inglês) Os dados se repetem 3 vezes para definir as permissões do usuário, do grupo e de outros. |
| 6 | O diretório X11R6 possui 6 links. |
| root | Usuário root é dono do diretório X11R6 |

| | |
|--------------|--|
| root | Grupo root é dono do diretório X11R6 |
| 4096 | O diretório X11R6 tem o tamanho de 4096 bytes. |
| Jan 11 06:23 | O diretório X11R6 foi modificado pela última vez em 11 de Janeiro às 06:23 horas. |
| X11R6 | Nome do diretório. |

Para mostrar um exemplo diferenciado, vamos utilizar como exemplo a linha 5 da saída.

| | |
|---|--|
| lrwxrwxrwx 1 root root 10 Jan 11 09:41 info -> share/info | |
| lrwxrwxrwx | 1 significa link |
| 1 | O link info possui 1 outro link para ele. |
| root | Usuário root é dono do link info |
| root | Grupo root é dono do link info |
| 10 | O link info tem o tamanho de 10 bytes. |
| Jan 11 09:41 | O link info foi modificado pela última vez em 11 de Janeiro às 09:41 horas. |
| info -> share/info | O link info aponta para o path share/info (path relativo). Links serão explicados detalhadamente no capítulo 4. Mas este é um bom exemplo de como identificar um link; sempre após um link aparecem os caracteres “->” simbolizando uma seta que aponta para um path ou arquivo. |

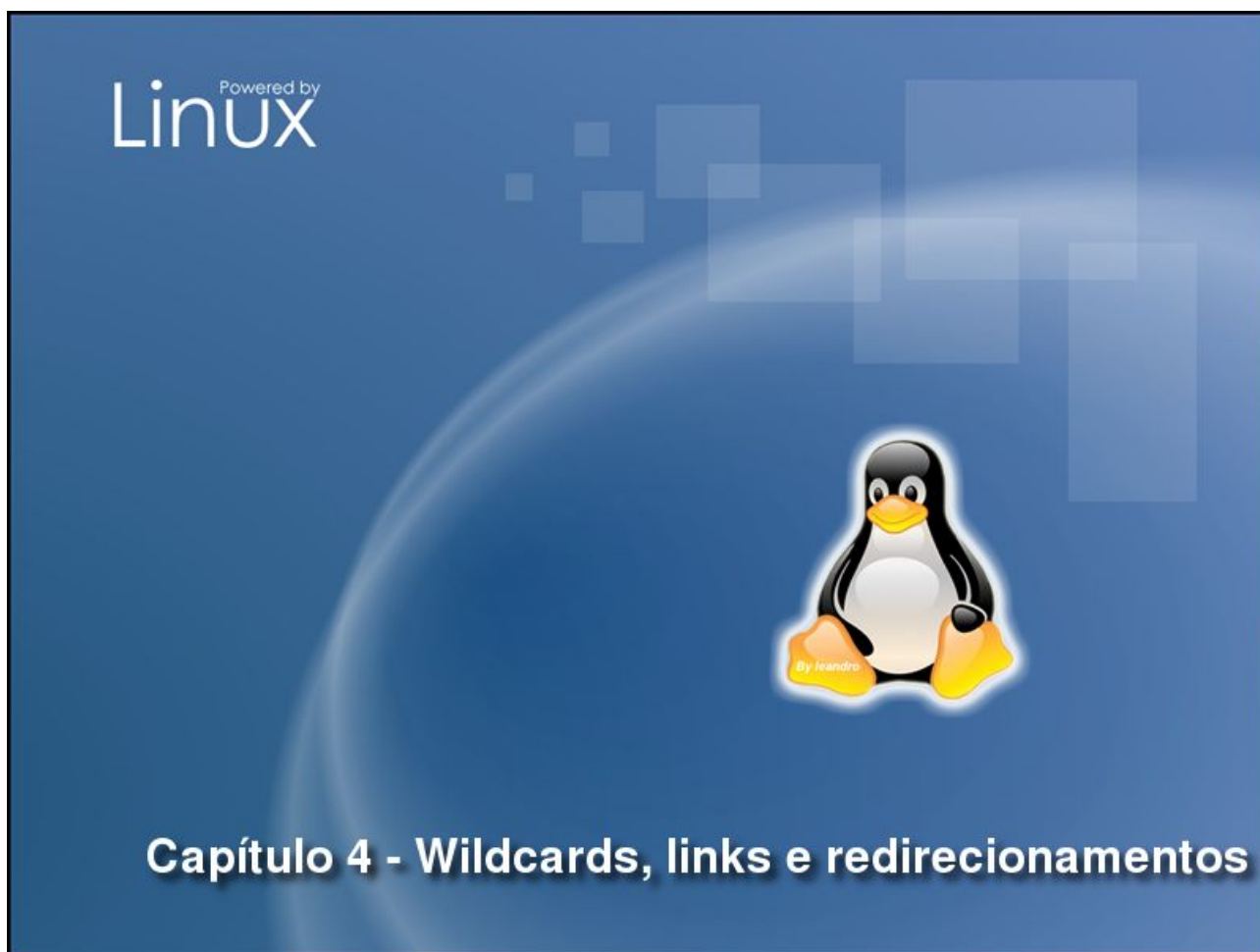
O comando `ls` tem diversas opções, para uma olhada detalhada recomendo que leia o manual deste comando.



Exercícios de fixação

Para fixar o conteúdo aprendido execute os exercícios abaixo:

- Consulte o manual e as principais opções dos comandos abaixo:
 - `touch`
 - `mkdir`
 - `mv`
 - `cp`
 - `rm`
 - `rmdir`
 - `ls`
- Crie o diretório **exercicioaula3/**, copie o conteúdo do diretório **/aulas/aula3/** para este diretório, execute o comando `ls -l exercicioaula3/`.
- Remova o diretório **exercicioaula3/**.



No capítulo anterior vimos os principais comando de manuseio de arquivos no linux, vimos também como consultar os manuais de comando, de agora em diante os comandos na apostila serão somente mencionados, não serão explicados como anteriormente, pois podem ser consultados nos manuais. Vamos agora aprender Wildcards que são caracteres especiais para formatação de comandos, isto nos permite selecionar um ou mais arquivos por vez nas nossas saídas ou então criar exceções. Aprenderemos também como criar links para arquivos e diretórios, e como manusear estes links. Por fim, vamos aprender a fazer o redirecionamento de comandos no GNU/Linux. Ao final deste capítulo você será capaz de:

- Customizar a saída de comandos com caracteres especiais
- Executar as tarefas das aulas anteriores com maior rapidez
- Criar links
- Distinguir soft links e hard links
- Fazer redirecionamento de outputs de comandos para outros comandos.



4.1 Wildcards

Wildcards servem basicamente para substituir caracteres são conhecidos como caracteres curinga, na sua utilização do Linux, você vai notar que quando precisar executar uma simples operação como a remoção de diversos arquivos de uma vez, pode tornar-se uma tarefa repetitiva e lenta.

```
$ rm laranja1 laranja2 laranja3 laranja4 laranja5 laranja6
```

Para resolver este problema, o Linux possui suporte a wildcards, também conhecido como “glob” (man glob). Para então deletarmos os arquivos acima podemos utilizar a sintaxe abaixo:

```
$ rm laranja[1-6]
```

Isto vai apagar os arquivos **laranja** que vão de 1 a 6, pode ser também que hajam mais arquivos **laranja** depois do 6 que também devem ser deletados. neste caso utilizamos a sintaxe abaixo:

```
$ rm laranja*
```

Isto vai deletar todos os arquivos que começam com a palavra laranja, veja bem que o * substitui tanto números quanto letras. Abaixo segue mais um exemplo de utilização, vamos listar todos os arquivos no diretório /etc que começam com a letra c:

```
$ ls -d /etc/i*
/etc/identd.conf      /etc/imlib           /etc/inittab         /etc/issue
/etc/identd.key       /etc/inetd.conf      /etc/inputrc         /etc/issue.net
/etc/idle-python2.1   /etc/init.d          /etc/irda.conf
```

4.1.1. Sintaxe “ * ”

O coringa * substitui zero ou mais caracteres, isso significa “qualquer coisa pode vir aqui, incluindo nada”.

Exemplos:

- /etc/g* equivale a todos os arquivos dentro do diretório **/etc/** que começam com g ou são chamados g.
- /tmp/my*1 equivale a todos os arquivos dentro do diretório **/tmp/** que começam com my e terminam com 1.

4.1.2. Sintaxe “ ? ”

O coringa ? substitui um único caractere.

Exemplos:

- myfile? equivale a todos os arquivos que começam com myfile e são seguidos por um único caractere.

- /tmp/notes?txt pode ser tanto /tmp/notes.txt quanto /tmp/notes_txt se existirem, mas não podem ser /tmp/notes.atxt pois este “a” não é substituído.

4.1.3. Sintaxe “ [] ”

O coringa [] é semelhante ao ?, porém este permite maior precisão, o funcionamento deste wildcard consiste em colocar os caracteres procurados dentro dos colchetes, você pode inclusive especificar um conjunto de caracteres em ordem.

Exemplos:

- myfile[12] equivale a myfile1 e myfile2, não a myfile12 pois o wildcard [] substitui um único caractere.
- [Cc]hange[Ll]og equivale a Changelog, ChangeLog, changeLog e changelog. Como você pode ver a utilização do wildcard colchetes é útil na verificação de letras maiúsculas e minúsculas.
- ls /etc/[0-9]* vai listar todos os arquivos no diretório /etc/ que começam com um número.
- ls /tmp/[A-Za-z]* vai listar todos os arquivos no diretório /tmp que começam com uma letra maiúscula ou minúscula.

4.1.4. Sintaxe “ [!] ”

O coringa [!] é semelhante ao [], porém este explicita exceções, ou seja tudo o que não estiver dentro dos colchetes.

- rm myfile[!9] vai remover todos os arquivos que começam com myfile mais um único caractere exceto o arquivo myfile9.

4.1.5. Formatação de wildcards

Por serem caracteres especiais, os caracteres wildcards (?, [,], !, *) são tratados pelo bash de uma maneira diferenciada, desta forma devemos tomar alguns cuidados quando escrevemos estes caracteres no bash. As três formas mais comuns de se escrever um caractere wildcard em sua forma literal (por exemplo; fazer o bash entender o ? como um ponto de interrogação, e não como qualquer coisa como vimos anteriormente) devemos colocar os caracteres em uma das maneiras abaixo:

- Entre aspas simples: '?'
- Entre aspas duplas: “?”
- Com uma barra invertida na frente do caractere: \?

Observação: as aspas simples funcionam de maneira semelhante a aspas duplas, porém as aspas simples permitem ao bash a execução de código.



4.2 Links

Links, em inglês significa ligação, no nosso caso são atalhos, servem principalmente para criar alternativas de paths para arquivos e programas. Imagine por exemplo um arquivo de texto localizado no diretório, /usr/local/share/documentos/ com o nome de texto.txt, para editarmos este arquivo usamos o editor Vi, então sempre que formos abrir este texto deveremos digitar no console:

```
$ vi /usr/local/share/documentos/texto.txt
```

E um caminho grande e de difícil memorização concorda? Agora se criarmos um link nossa pasta home, ficaria muito mais fácil pois sempre que formos abrir este texto pelo atalho basta digitar:

```
$ vi ~/texto.txt
```

Um outro exemplo, temos um programa cujo path é /usr/local/share/exemplo, ao invés de abrir o programa por seu path absoluto, podemos criar um link para o mesmo no diretório /usr/local/bin, que é um diretório padrão para links de programas. Desta forma para abrimos o tal programa podemos simplesmente digitar seu nome no console.

Existem dois tipos de links, os soft links e os hard links, a funcionalidade dos dois é praticamente a mesma, ambos são caminhos alternativos para determinado arquivo, no entanto, os dois tem diferenças radicais entre si.

4.2.1. Soft links

São os links mais comuns, também chamados de symbolic links, (links simbólicos), são tipos de links que se referem a arquivos pelo seu nome, se o arquivo destino for movido ou deletado o link perde sua funcionalidade se tornando um broken link (link quebrado).

A criação de links simbólicos é feita utilizando o comando `ln` com a opção `-s`:

```
$ ln -s /aulas/aula4/links1.txt ~/
```

Agora o arquivo de texto links1.txt tem um link na sua home, experimente agora utilizar o comando `ls -l` e veja o output para o link e confirme o caminho do mesmo:

```
lrwxr-xr-x 1 bruno users 46 May 11 09:59 links1.txt -> /aulas/aula4/links1.txt
```

Veja que o output confirma que nosso link esta para o caminho correto.

Agora vamos tentar criar um link simbólico de uma outra forma, vamos para o diretório /aulas/aula4/, criaremos o link e moveremos o link para a home com o comando `mv`.

```
$ cd /aulas/aula4/
$ ln -s links2.txt ln2.txt
```

Criamos o link para o arquivo links2.txt com o nome ln2.txt, agora verifique para onde aponta o link ln2.txt:

```
lrwxr-xr-x 1 bruno users 46 May 11 10:14 ln2.txt -> links2.txt
```

Veja que o link aponta para o path relativo, experimente agora mover o link ln2.txt para a sua home, logo após tente abrir o arquivo pelo link ln2.txt com o programa Vi:

```
$ mv ln2.txt ~/
$ cd ~/
$ vi ln2.txt
```

Veja que o link não funciona. Explicação; observando novamente para onde apontamos o link ln2.txt, vemos que o mesmo aponta para o arquivo links2.txt, até aí nenhum segredo, mas não tem nenhum arquivo links2.txt no diretório home, ou seja não vai funcionar pois na saída contém o path relativo do arquivo de destino, o que não funciona em todas as ocasiões. Portanto aqui vai a lição mais importante de links simbólicos; analise quando utilizar paths absolutos e paths relativos de links, veja qual é mais vantajoso em cada ocasião.

4.2.2. Hard links

Estes links funcionam de maneira diferente, ao invés de se basearem em nomes e paths, eles se baseiam diretamente no hardware para criar o link. No hardware? De que maneira? Aqui vai uma breve explicação:

Qualquer objeto localizado no disco rígido tem um número único de índice (index) chamado de inodo, os hard links se baseiam neste número, ao invés do nome do arquivo propriamente dito, ou seja ele cria um atalho para a própria localização no disco. Um inodo pode então conter um ou mais hard links, e vai existir até que não existam mais links para ele.

Copie o arquivo hardlink.txt do diretório /aulas/aula4/ para sua home e utilize o comando `ls -i | grep hardlink.txt` (este último comando será explicado nos próximos capítulos, se chama redirecionamento).

```
$ cp /aulas/aula4/hardlink.txt ~/
$ ls -i | grep hardlink
2073563 hardlink.txt
```

Pronto, obtivemos o número do inodo do arquivo hardlink.txt, (comando `ls -i`), diante disso criaremos um hard link com o comando `ln` sem parâmetros para o arquivo hardlink.txt:

```
$ ln hardlink.txt hardlink2.txt
```

Com o hard link criado, veja agora os números de inodos dos arquivos:

```
$ ls -i | grep hardlink
2073563 hardlink.txt
2073563 hardlink2.txt
```

Como você pode verificar, os números de inodo são exatamente iguais, experimente agora verificar se são links pelo comando `ls -l | grep hardlink`:

```
$ ls -l | grep hardlink
-rw-r--r--  2 d3v11 users      39 May 11 10:44 hardlink.txt
-rw-r--r--  2 d3v11 users      39 May 11 10:44 hardlink2.txt
```

Como você pode ver, os links não aparecem nem com o `l` nos triplets, nem com o `->` path, este é um detalhe o qual devemos estar atentos.

Experimente agora remover o arquivo original de destino, ou seja o `hardlink.txt` e tente abrir o link `hardlink2.txt` com o Vi:

```
$ rm hardlink.txt
$ vi hardlink2.txt
```

```
Este sera um destino para o hard-link.
```

```
~
~
~
```

```
"hardlink2.txt" 1L, 39C
```

```
1,1
```

```
All
```

Para a nossa surpresa o link funcionou, isto comprova que o link não está vinculado ao arquivo de destino e sim ao inodo.

Links utilizando hard links só podem ser feitos para arquivo, e não para diretórios, na verdade os atalhos `.` e `..` são hard links para diretórios mas estes foram criados pelo sistema operacional, e não por um usuário, nem sequer o root pode criar hard links para diretórios. Outra limitação, os hard links não podem transpor sistemas de arquivo diferentes.



4.3 Redirecionamento

O bash nos permite redirecionar outputs de comandos para outros comandos ou para arquivos, nos proporcionando grandes possibilidades se somadas aos inúmeros comandos do GNU/Linux.

4.3.1. Pipe

O pipe "`|`" serve para passar o output de um comando para outro, ou seja antes de passar para a tela o resultado você pode literalmente redirecionar para onde vai seu comando, veja a sintaxe abaixo:

```
$ ls /etc/ | grep conf | sort -r | lpr -p LaserJet
```

A sintaxe acima tem 3 redirecionamentos chamados Pipes, o que os comandos acima fazem; Primeiro o comando `ls` lista tudo o que está no diretório `/usr/share/empresas/`, o **pipe** manda essa listagem para o comando `grep` que discarta tudo o que não tem a palavra **conf**, o terceiro **pipe** manda esta listagem filtrada para o comando `sort -r` que coloca tudo em ordem decrescente, e por último o terceiro **pipe** manda o resultado para o comando `lpr -p LaserJet` que imprime a listagem filtrada e ordenada. Interessante não? O exemplo acima pode parecer um pouco complicado mas vamos agora para um comando mais simples:

```
$ cat /etc/samba/smb.conf | mail brunocesar@ajato.com.br
```

O **cat** mostra o conteúdo do arquivo **smb.conf**, o **pipe** manda este conteúdo para o comando **mail** que manda um e-mail para brunocesar@ajato.com.br. Pipes são úteis também quando o output de um comando é muito grande, veja so:

```
$ ls /etc/ | grep conf | less
```

O **ls** lista o conteúdo do diretório **/etc/**, o **pipe** manda esta lista para o comando **grep** que filtra os arquivos com a palavra **conf** e finalmente o último **pipe** manda a lista para o comando **less** que é um programa pager, ou seja, exibe as paginas permitindo exibir o inicio do texto com a tecla seta p/ cima e ler tudo paulatinamente.

4.3.2. >

Um comando seguido pelo símbolo “>” redireciona seu output para um arquivo, utilizarei o mesmo exemplo anterior:

```
$ ls /etc/ | grep conf > novo.txt
```

O **ls** lista o conteúdo do diretório **/etc/**, o **pipe** manda esta lista para o comando **grep** que filtra os arquivos com a palavra **conf**, o **>** manda esta lista para o arquivo **novo.txt**, para comprovar isto exiba o conteúdo do arquivo **novo.txt** com o comando **cat novo.txt | less**.

Uma observação que deve ser feita: se o arquivo de destino existir ele será sobrescrito por este redirecionamento, ou seja o conteúdo anterior do arquivo será substituído pelo novo.

4.3.3. >>

O símbolo “>>” funciona de maneira semelhante ao símbolo “>” mas este por sua vez não substitui o conteúdo do arquivo de destino (caso ele exista), somente adiciona ao final do arquivo. Vamos exemplificar:

```
$ echo "Ola mundo">arquivo.txt
$ cat arquivo.txt
Ola mundo
```

O vimos que o redirecionamento funcionou, agora vamos adicionar a String “tudo bem?” no arquivo:

```
$ echo "tudo bem?">arquivo.txt
$ cat arquivo.txt
tudo bem?
```

Peraí, cade o “Ola mundo” ? Resposta: foi apagado porque o redirecionamento utilizado foi o “>”. Veja agora como seria o correto:

```
$ echo "Ola mundo">arquivo.txt
$ cat arquivo.txt
Ola mundo
$ echo "tudo bem?">>arquivo.txt
$ cat arquivo.txt
Ola mundo
tudo bem?
```

Pronto, pode parecer um exemplo bobo mas imagine se fosse uma rotina de logs, de sistema.

4.3.4. <<

O Bash e os outros shells suportam o conceito de “arquivoaqui”, ou seja ele pode interpretar que o próprio console e um arquivo se desejar, podendo estabelecer variáveis entre outras opções. Isso permite estabelecer um input para um comando em varias linhas, vejamos o exemplo:

```
$ sort <<CHAMADA
Tadeu
Jorge
Bruno
Renato
Marcos
CHAMADA

Bruno
Jorge
Marcos
Renato
Tadeu
```

Ou seja, o comando `sort`, recebeu do redirecionamento `<<` uma lista de nomes e colocou os mesmos em ordem alfabética. A lista no caso era uma variável, agora voce pode ver como o shell é extremamente poderoso.

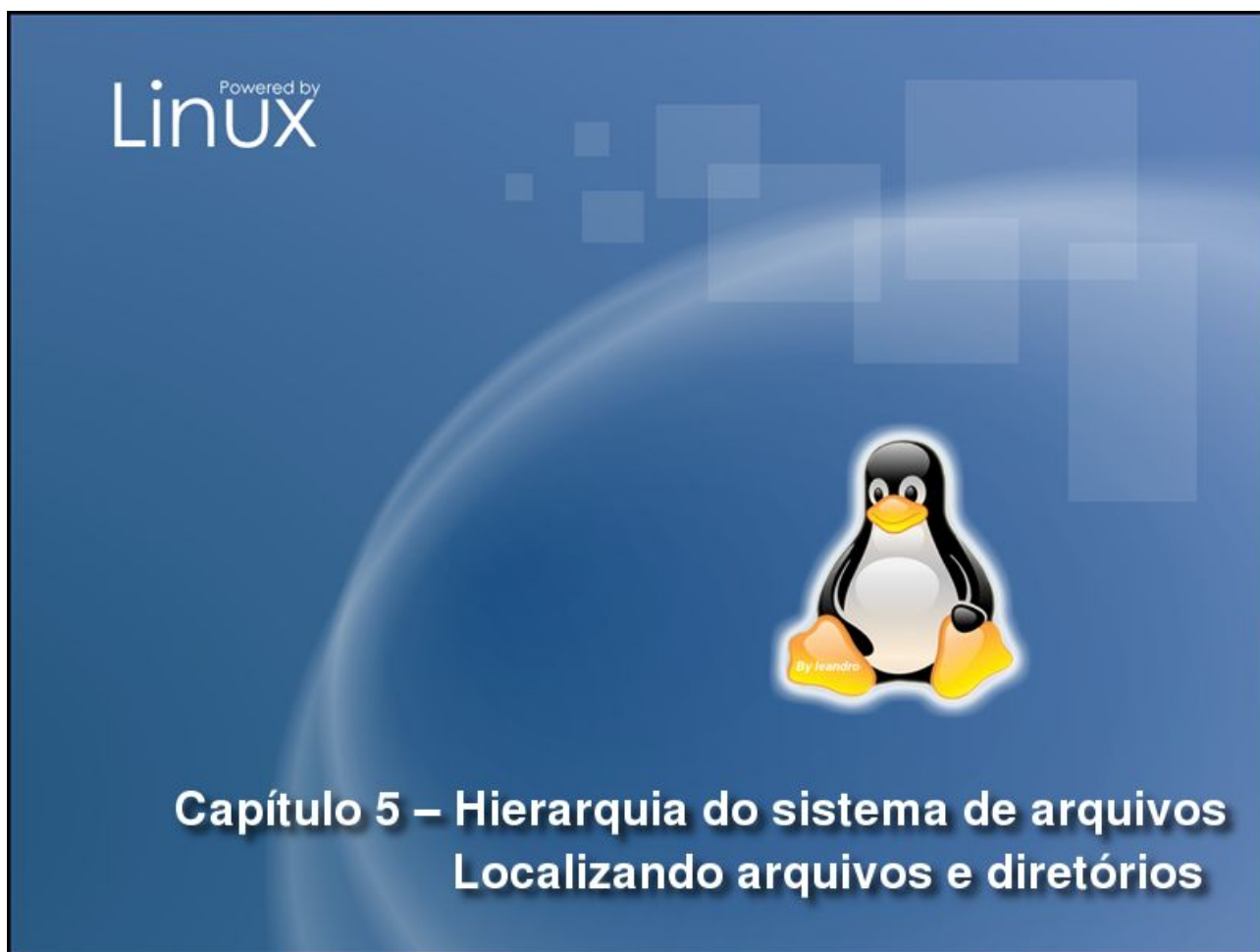


Exercícios de fixação

1. Procure utilizando wildcards:
 - a. Todos os arquivos no diretorio `/etc` que começam com **g** e terminam com **.conf**
 - b. Todos os arquivos no diretorio `/etc` que contém um numero qualquer
 - c. Todos os arquivos no diretorio `/etc` que começam com **a, b, c, d, e** ou **f** e terminam com **.conf**
 - d. Todos os arquivos no diretorio `/etc` que não começam com **g** e terminam com **.conf**
2. Crie um arquivo vazio com o nome de `arquivteste.txt`, depois crie um hard link para o

arquivoteste.txt com o nome de hardlinkteste.txt e finalmente crie um link simbólico para o hardlinkteste.txt com o nome de softlinkteste.txt1

3. Verifique se pode identificar hardlinks e softlinks.
4. Treine os redirecionamentos |, >, >> e <<.



No capítulo anterior você aprendeu sobre wildcards, links e redirecionamento de outputs. Neste capítulo vamos deixar o shell um pouco de lado e verificar a FHS (Filesystem Hierarchy Standard) ou seja, a hierarquia do sistema de arquivos, que nada mais é do que a padronização de uma disposição com que os diretórios são colocados independentemente de distribuição. Você também vai aprender como localizar arquivos e diretórios utilizando os diversos comandos para esta finalidade. Ao final deste capítulo você será capaz de:

- Dominar a árvore de diretórios do GNU/Linux
- Localizar arquivos corretamente



5.1 Hierarquia do sistema de arquivos

Apesar de parecer confusa no início, a árvore de diretórios do GNU/Linux é muito bem organizada e é respeitada em praticamente todas as distribuições, podem haver pequenas variações, encontramos os seguintes diretórios:

- **/** (diretório root)

Abaixo deste estão:

- **/bin** (diferente do **/sbin**, este diretório contém vários comandos úteis utilizados tanto para o administrador do sistema quanto para o usuário comum, contém geralmente os shells, e os programas mais comuns)
- **/boot** (diretório onde ficam os arquivos essenciais para o boot do GNU/Linux, fica também os novos kernels recompilados e reinstalados.)
- **/dev** (diretório onde ficam armazenados arquivos especiais ou arquivos de dispositivos, o GNU/Linux entende todos os dispositivos como arquivos, tanto o seu dispositivo usb quanto sua placa de som estão aqui.)
- **/etc** (diretório com os arquivos de configuração de programas, serviços ou do próprio Linux, é este o diretório em que devem ser procurados os arquivos de configuração)
- **/home** (o Linux é um ambiente multi usuário, o que implica que cada usuário tem um diretório que pode ser acessado por si e pelo administrador do sistema, estes são os diretórios home e são acessíveis por **/home/Nomedousuario**)
- **/lib** (diretório onde são encontradas as bibliotecas compartilhadas de aplicativos e módulos do kernel)
- **/lost+found** (em sistemas de arquivos com a capacidade de journaling este diretório armazena os dados recuperados pelo fsck)
- **/mnt** (diretório temporário que serve de ponto de montagem para dispositivos de armazenamento, tornando-os acessíveis)
- **/opt** (diretório reservado para todos os softwares que não fazem parte da instalação padrão)
- **/proc** (diretório com sistema de arquivos virtual do kernel que demonstra status de programas ou então informações sobre o hardware, e até mesmo a alteração do status de hardware, não contém arquivos reais)
- **/root** (diretório home do usuário root)
- **/sbin** (diretório com arquivos executáveis de manutenção do sistema e de tarefas administrativas)
- **/tmp** (diretório com arquivos temporários, muitos programas utilizam-se deste diretório para criar arquivos lock, e também armazenamento de dados temporário)
- **/usr** (contém uma outra árvore de diretórios compartilhada entre os usuários, bibliotecas, aplicativos, documentação e etc.)
- **/var** (contém dados variáveis como logs de arquivos e spools de impressoras)

Não há muito o que discutir à respeito do FHS, mais informações à respeito e

diversos exemplos podem ser encontrados em <http://www.pathname.com/fhs/>



5.2 Localizando arquivos

Um sistema Linux comum instalado possui centenas de milhares de arquivos, para auxiliar no gerenciamento desses arquivos todos temos diversas ferramentas de busca no Linux, cada qual com suas vantagens e desvantagens, cada uma se adapta a uma situação.

5.2.1. O \$PATH

Esta é uma variável padrão do bash, como disse anteriormente o bash pode armazenar variáveis de ambiente (variáveis de ambiente sempre terão o símbolo \$ no início da palavra), esta por exemplo armazena caminhos de programas binários. Quando você executa um comando, o bash procura automaticamente este comando em um dos diretórios armazenados na variável \$PATH. Veja o exemplo abaixo:

```
$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:/usr/bin/
```

Os diretórios estão separados por dois pontos (:), quando executarmos um comando qualquer o bash irá procurar este comando em um dos diretórios listados acima. Nos próximos capítulos aprenderemos mais sobre variáveis de ambiente.

5.2.2. Comando whereis

O comando `whereis` procura na variável \$PATH o arquivo, o retorno é instantâneo porém só retorna programas, ou seja se você estiver procurando um programa utilize o `whereis`. Exemplo:

```
$ whereis gzip
gzip: /bin/gzip /usr/share/man/man1/gzip.1.gz
```

O output do comando foram dois diretórios, o primeiro é o local onde se encontra o programa `gzip`, o segundo é o local onde se encontra a página de manual do `gzip`, ou seja; `man gzip`. Geralmente os outputs do comando `whereis` seguirão esta ordem.

5.2.3. Comando locate

O comando `locate` funciona de outra maneira, ele retorna os resultados da pesquisa instantaneamente também, sua fonte de pesquisa no entanto é um banco de dados de arquivos criado pelo Linux, este banco de dados pode ser atualizado utilizando o comando `updatedb`, sua vantagem é que pode retornar qualquer tipo de arquivo e diretório, não tendo restrições em suas buscas, outra opção interessante é a possibilidade de utilização de wildcards em suas buscas. Exemplo:

```
$ locate firefox
/usr/bin/firefox
/usr/bin/mozilla-firefox
/usr/lib/menu/mozilla-firefox
/usr/lib/mime/packages/mozilla-firefox
```

Geralmente o output do comando `locate` é muito grande, não listei todos. Sua desvantagem é que o banco de dados pode ficar desatualizado em poucos dias, (quando o banco tem mais de 8 dias o comando `locate` já avisa) e sua atualização demora algum tempo, pode demorar alguns minutos dependendo da velocidade de seu HD, esta atualização só pode ser efetuada pelo usuário `root`.

```
# updatedb
```

Depois de aguardar alguns minutos o novo banco de dados estará atualizado, esta demora ocorre pois o sistema vai reindexar todos os arquivos no banco de dados.

5.2.4. Comando `find`

O comando `find` não tem uma fonte de pesquisa, na verdade sua fonte de pesquisa vai ser passada como parâmetro no comando, desta forma ele pode ser usado em qualquer situação, pode localizar arquivo, diretórios, textos dentro de arquivos, arquivos por tempo, por tipo de arquivo e também por seu tamanho. Como podemos ver é uma busca bem completa, sua desvantagem é o tempo necessário para a pesquisa, por ser feita em tempo real pode demorar desde segundos até alguns minutos, tudo vai depender dos parâmetros passados na pesquisa. Veja o exemplo abaixo:

```
$ find /usr/share/doc -name README\*
/usr/share/doc/base-files/README.base
/usr/share/doc/base-passwd/README
/usr/share/doc/bash/README.Debian.gz
/usr/share/doc/bash/README.bash_completion.gz
/usr/share/doc/bash/README.abs-guide
/usr/share/doc/bash/README.commands.gz
```

Comando: `find </diretorio/> -opcao <parametro>`

No caso utilizamos a opção `-name` que especifica o nome do arquivo.

O output deste comando também foi volumoso, portanto retirei diversos resultados. A procura acima consistia em procurar no diretório `/usr/share/doc/` todos os arquivos que tinham como nome a palavra **README+qualquercoisa**, (lembra-se que o `*` significa qualquer coisa certo?). Para ignorar letras maiúsculas e minúsculas deve-se colocar como opção `-iname`.

Para procurar agora por expressões regulares, (veremos expressões regulares no próximo capítulo), serve para procurar arquivos e tendo como critério o que está dentro deste arquivo. um exemplo; para procurar no diretório `/etc/` todos os arquivos que contêm

a expressão “php” utilizamos a seguinte sintaxe:

```
$ find /etc/ -regex ".*php.*"  
/etc/cron.d/php4  
/etc/php4  
/etc/php4/apache  
/etc/php4/apache/php.ini  
/etc/apache/conf.d/phpmyadmin.conf  
/etc/apache/conf.d/php4.conf  
/etc/apache-perl/conf.d/phpmyadmin.conf  
/etc/apache-ssl/conf.d/phpmyadmin.conf  
/etc/phpmyadmin  
/etc/phpmyadmin/apache.conf  
/etc/phpmyadmin/config.footer.inc.php  
/etc/phpmyadmin/config.header.inc.php  
/etc/phpmyadmin/config.inc.php  
/etc/phpmyadmin/htaccess  
/etc/phpmyadmin/blowfish_secret.inc.php
```

Utilizamos a opção `-regex` que especifica uma expressão regular dentro de um arquivo (simplificando, um texto, uma frase uma palavra escrita em alguma parte do arquivo), para ignorar diferenciação entre maiúsculas e minúsculas utilize a opção `-iregex`. Existem muitas outras opções no comando `find`, o manual deste comando contém explicações e exemplos sobre estas opções .



Exercícios

1. Procure o arquivo `passwd` em todo o sistema operacional.
2. Procure o programa `xcalc`.
3. Procure o arquivo `smb.conf` dentro do diretório `/etc/` .



No capítulo anterior vimos o FHS do Linux e aprendemos como localizar arquivos e diretórios. Neste capítulo iremos aprender expressões regulares e também como controlar os processos de programas. Ao final deste capítulo você será capaz de:

- Entender a sintaxe de regex (regular expressions)
- Abrir programas em foreground e background
- Modificar estado de execução de programas
- Fechar programas travados
- Listar e gerenciar processos pelo PID
- Definir prioridades de processos



6.1. Expressões regulares.

Expressões regulares, ou “regex” são sintaxes usadas para descrever textos. No Linux, são utilizadas principalmente para encontrar textos em arquivos. Funcionam de maneira semelhante a dos wildcards, ou seja funcionam como caracteres especiais que facilitam a busca de informações.

6.1.1. Texto simples

Um texto simples é considerado como um regex básico, por exemplo se desejamos procurar em um arquivo uma expressão de texto qualquer utilizamos o comando `grep`, o comando `grep` imprime todas as linhas que contém a regex passada como parâmetro. Veja o exemplo abaixo:

```
$ grep ftp /etc/services
ftp-data      20/tcp
ftp           21/tcp
tftp          69/udp
sftp          115/tcp
ftps-data     989/tcp      # FTP over SSL (data)
ftps          990/tcp
venus-se      2431/udp     # udp sftp side effect
codasrv-se    2433/udp     # udp sftp side effect
frox          2121/tcp     # frox: caching ftp
proxy
zope-ftp      8021/tcp     # zope management by ftp
```

O comando `grep` procurou pela regex **ftp** dentro do arquivo **/etc/services** e imprimiu somente as linhas que continham esta regex. Dica; utilize sempre aspas em regex pois existem alguns caracteres especiais que podem alterar o resultado da procura.

6.1.2. Sintaxe “.”

O “.” em uma regex trabalha de maneira semelhante ao “?” nos wildcards, ou seja substitui um único caractere qualquer, exemplo:

```
$ grep "dev/hda." /etc/fstab
/dev/hda2    /          ext3      errors=remount-ro    0        1
/dev/hda3    none       swap      sw                   0        0
/dev/hda1    /mnt/windows ntfs      rw,user,exec,noauto  0        0
```

6.1.3. Sintaxe “[]”

A sintaxe “[]” é utilizada quando se quer restringir a busca do “.”, especificando entre colchetes os caracteres que deverão retornar, veja o exemplo:

```
$ grep "dev/hda[23]" /etc/fstab
/dev/hda2      /                ext3      errors=remount-ro    0        1
/dev/hda3      none            swap      sw                0        0
```

Lembre-se que quando usados os [] sempre será substituído somente um caractere.

6.1.4. Sintaxe “ [^] ”

A sintaxe “ [^] ” realiza o trabalho inverso aos “ [] ”, ou seja, vai retornar as expressões que não contenham o que esta dentro de [^], veja o exemplo:

```
$ grep "dev/hda[^23]" /etc/fstab
/dev/hda1      /mnt/windows    ntfs      rw,user,exec,noauto 0        0
```

6.1.5. Sintaxe “ * ”

A sintaxe “ * ” serve para modificar o significado da regex anterior, dizendo que a regex anterior pode ocorrer zero ou mais vezes repetidas, (vimos ate agora que só e substituído um caractere por vez).

```
$ grep ".*bash" /etc/shells
/bin/bash
/bin/rbash
```

O comando `grep` imprimiu as linhas que continham um caractere qualquer, repetido diversas vezes e que era seguido do texto simples **bash**.

6.1.6. Sintaxe “ ^ ” e “ \$ ”

Os caracteres “ ^ ” e “ \$ ” servem respectivamente para localizar os caracteres do inicio e do final de uma linha. Utilizando a regex “ ^ ” especificamos o caractere inicial de uma linha, veja o exemplo:

```
$ grep ^# /etc/fstab
# /etc/fstab: static file system information.
#
# <file system> <mount point>    <type>    <options>                <dump>
```

Já a regex “ \$ ” serve para verificar o caractere final de uma linha:

```
$ grep '\.$' /etc/fstab
# /etc/fstab: static file system information.
```




6.2 Controle de processos

O kernel do Linux controla aplicativos que estão no nível do usuário através de processos, organizados por seus PIDs (process identification). Os processos são gerenciados automaticamente pelo kernel, porém, o usuário também pode gerenciar estes processos conforme sua necessidade. Antes de mais nada para iniciarmos um processo precisamos iniciar um aplicativo.

6.2.1. Iniciando um aplicativo

Para acompanharmos um processo corretamente à nível didático, iniciaremos um programa através de um shell comum, primeiro abra um console e inicie um programa qualquer, neste exemplo utilizaremos o programa xeyes:

```
$ xeyes
```

Imediatamente o programa vai iniciar, são dois olhos que ficam seguindo o mouse, note que o terminal ficou bloqueado, não aceitando mais nenhum comando.

6.2.2. Parando um aplicativo

Aplicativos podem ser parados a qualquer momento de sua execução, quando paramos um aplicativo este vai parar de consumir processamento mas ainda vai consumir memória pois não foi finalizado, todos os aplicativos parados ficarão estáticos mas ainda estão abertos e funcionando. A maneira mais prática de se parar um aplicativo é apertando as teclas Ctrl + Z no shell onde se iniciou o aplicativo:

```
[1]+  Stopped                  xeyes
$
```

Note agora que o programa parou de funcionar, como se estivesse travado (os olhos não seguem mais o movimento do mouse). Note também que o shell ficou livre novamente para a execução de comandos. Uma outra maneira de parar um processo, que vai funcionar em todos os casos por sinal, é executando o comando `kill -STOP <pid>`, ou seja o comando `kill` serve para enviar comandos para processos, a opção `-STOP` sinaliza que o comando `kill` deve parar processo e o parâmetro `pid` é o número `pid` do processo que veremos como obter adiante.

6.2.3. Listando processos

Para listarmos os processos no Linux utilizamos o comando `ps`, assim podemos obter os pids de processos e saber quais e quantos processos estão sendo executados no momento, para vermos os processos sendo executados no bash atual utilizamos o comando `ps` sem parâmetros exemplo:


```
$ ps
  PID TTY          TIME CMD
 9598 pts/4    00:00:00 bash
 9978 pts/4    00:00:00 xeyes
10130 pts/4    00:00:00 ps
```

Abaixo a descrição das colunas da lista obtida:

| | |
|------|---|
| PID | Número de identificação do processo |
| TTY | Console sob o qual o processo esta sendo executado |
| TIME | Tempo de processamento na CPU formatado em HH:MM:SS |
| CMD | Comando ou aplicativo |

Para obtermos uma listagem mais completa utilizamos a opção “ps u” veja abaixo:

```
$ ps u
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
d3v11        4303  0.0  0.6   2692   1564 tty2    Ss+  07:26   0:00 -bash
d3v11        9598  0.0  0.6   2668   1488 pts/4    Ss   09:59   0:00 /
bin/bash
d3v11        9978  0.0  0.6   3152   1504 pts/4    T    10:22   0:00 xeyes
d3v11       10536  0.0  0.3   2500    860 pts/4    R+   10:49   0:00 ps u
```

Descrição:

| | |
|------|--|
| USER | Usuário que executou o processo |
| %CPU | Porcentagem de utilização do processador |
| %MEM | Porcentagem de utilização de memória |
| VSZ | Tamanho de memória virtual utilizada |
| RSS | Espaço físico ocupado na memória pelo processo |
| STAT | São caracteres que identificam o estado do processo. (lista em man ps) |

Finalmente para listarmos todos os processos que estão sendo executados, (não os processos exclusivos do shell atual) utilizamos a opção “ps aux”. A lista é muito extensa portanto não será inclusa aqui.

Agora que temos conhecimento como trabalhar e identificar processos através do PID podemos passar para a próxima parte, vamos reiniciar o processo que paramos na seção anterior.

6.2.4. Foreground e Background

Foreground e background são os níveis de execução de aplicativos no sistema, foreground significa primeiro plano, e background significa plano de fundo ou segundo plano. Para reiniciarmos o nosso processo (xeyes) podemos utilizar duas opções, a primeira é utilizar o comando `fg` no shell onde o aplicativo foi iniciado/parado):

```
$fg  
xeyes
```

(A outra maneira seria utilizar o comando `kill -CONT <pid>`, a diferença é que o `-CONT` vai reiniciar o processo em background diretamente.)

O processo retornou à execução, porém o shell ficou bloqueado novamente. para mandarmos o processo para o background paramos novamente o processo e utilizamos o comando `bg` (o comando `kill -CONT <pid>` faz isto):

```
$ bg  
[2]+ xeyes &
```

Temos agora um bash funcional rodando um processo em background.

6.2.5. Finalizando um aplicativo

Para finalizarmos um aplicativo ou um programa travado utilizamos o comando `kill <pid>`, desta forma o programa vai ser fechado:

```
$ kill 9978  
[2]+ Terminated xeyes
```

O programa `xeyes` foi terminado. Se o programa estiver em foreground aperte as teclas `Ctrl+C`.

Uma outra maneira de finalizar programas pelo modo gráfico é utilizando o programa `xkill`, o mouse vai mudar de ícone e quando clicar na janela abaixo do mouse esta será fechada.

6.2.6. Iniciando um aplicativo em background

Para evitar todo este processo de iniciar o aplicativo, parar e enviar para background manualmente podemos simplesmente iniciar o processo em background, para isto utilizamos o caractere `&` logo após o comando. Veja o exemplo abaixo:

```
$ xeyes -center lightblue &  
[1] 12941
```

O programa `xeyes` foi iniciado agora em background, abaixo vemos dois campos, o primeiro é seu JOB number, ou seja é a tarefa número 1 do console atual, o segundo campo é seu PID. Podemos iniciar diversos aplicativos em background simultaneamente utilizando sempre o caractere `&` no final de cada comando.

6.2.7. Múltiplos processos em background

Vamos agora trabalhar com diversos processos, inicie em background o programa `xeyes` com a opção `-center pink`:

```
$ xeyes -center pink &  
[2] 13014
```

Veja que o JOB number mudou. Vamos agora listar os jobs do console atual com o comando `jobs -l`:

```
$ jobs -l  
[1]- 12941 Running                  xeyes -center lightblue &  
[2]+ 13014 Running                  xeyes -center pink &
```

O job 2 possui um sinal de + ao lado, isto significa que é o job atual e o comando `fg` vai surtir efeito diretamente neste e não no primeiro job (marcado com -). Caso queria mandar o primeiro para foreground especifique seu job number como parâmetro ao comando `fg`:

```
$ fg 1  
xeyes -center lightblue
```

6.2.8. Prioridades de processos

Os processos no Linux possuem prioridades que determinam como e com que frequência podem acessar à CPU. A maioria dos processos iniciados pelo usuário possuem a mesma prioridade (0), o que os torna iguais à nível de processamento. Imagine agora que você tenha que compilar um programa muito importante e ao mesmo tempo abre o seu player de mp3, os dois programas vão utilizar o processador da mesma forma e com a mesma frequência, seria mais interessante colocar o processo de compilação com uma prioridade maior, isto vai acelerar a compilação. Este é só um exemplo, um outro exemplo prático; na sua vida como administrador Linux haverá horas em que seu servidor de webpages está recebendo muitas requisições, nestas horas para o acesso do usuário não ficar lento você deverá também redefinir a prioridade do seu servidor. Para definirmos processos na hora em que iniciamos o programa utilizamos o comando `nice` antes do comando, veja o exemplo:

```
$ nice -n 10 mpg123 musica.mp3
```

Para o comando `nice`, quanto maior for o número passado como parâmetro, menor é sua prioridade, ou seja, no exemplo acima o programa `mpg123` vai ter uma prioridade menor do que a de um compilador por exemplo. O comando `nice`, no entanto, só pode definir a prioridade quando o processo é iniciado, para redefinir (modificar) a prioridade de algum processo no Linux você deverá utilizar o comando `renice`, veja no exemplo abaixo sua utilização:

```
$ nice -n 10 mpg123 musica.mp3
```

Para o comando `nice`, quanto maior for o número passado como parâmetro, menor é sua prioridade, ou seja, no exemplo acima o programa `mpg123` vai ter uma prioridade menor do que a de um compilador por exemplo. O comando `nice`, no entanto, só pode definir a prioridade quando o processo é iniciado, para redefinir (modificar) a prioridade de algum processo no Linux você deverá utilizar o comando `renice`, veja no exemplo abaixo sua utilização:

utilização:

```
$ firefox &
[1] 13627
$ renice 10 13627
13627: old priority 0, new priority 10
```

O processo `firefox` passou a ter a prioridade 10. Uma informação importante, quando um usuário comum tenta aumentar a prioridade de um processo ele é barrado pelo sistema, isto se dá por segurança, o Linux foi desenvolvido para ser um sistema multi-usuário ou seja diversos usuários ao mesmo tempo, seria uma bagunça tremenda se todos os usuários pudessem melhorar os desempenhos de seus processos em relação aos outros.

```
$ renice 0 13627
renice: 13627: setpriority: Permission denied
```

O único usuário que pode aumentar a prioridade de processos é o usuário `root`.

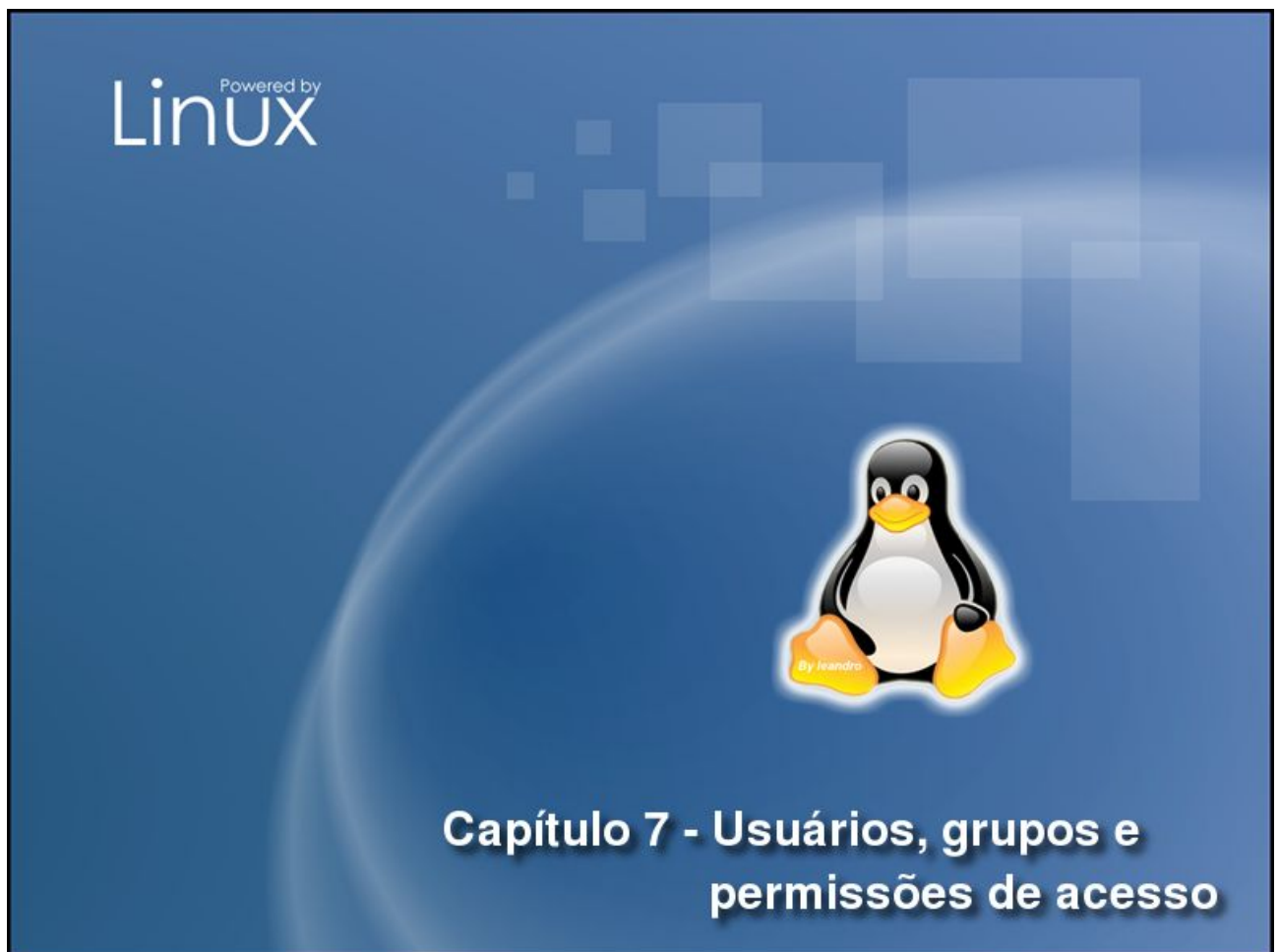
```
# renice 0 13627
13627: old priority 10, new priority 0
```

Para listar prioridades de processos digite o comando `ps -l` e acompanhe a coluna `NI`, lá está listada a prioridade de cada processo.



Exercícios

1. Exiba a regex **bash** dentro do arquivo **/etc/passwd**.
2. Exiba as entradas **/dev/hda1** e **/dev/hda3** do arquivo **/etc/fstab** utilizando um único comando.
3. Inicie um processo normal em foreground, mande o para background através de outro console, e redefina sua prioridade para 10, depois finalize o processo.



Neste capítulo conheceremos as características e ferramentas que tornam o GNU/Linux um sistema operacional multi-usuário, veremos como trabalhar com contas de usuários e grupos de usuários, também veremos como trabalhar com permissões de acesso a arquivos e diretórios e finalmente permissões especiais de arquivos e diretórios. Este capítulo é muito importante pois vai tornar clara como funciona a segurança com relação à usuários no Linux. Ao final deste capítulo você será capaz de:

- Adicionar, alterar e remover contas de usuários no sistema.
- Trabalhar com grupos de usuários.
- Entender e definir permissões de acesso à arquivos e diretórios.
- Entender e definir permissões de acesso especiais.



7.1. Usuários

Já foi comentado anteriormente que o Linux é um sistema multi usuário, mas afinal o que isto quer dizer? Um sistema operacional multi usuário permite que múltiplos usuários utilizem o computador e rodem programas ao mesmo tempo, compartilhando os recursos da máquina. Para que se utilize desta característica cada usuário deve ter uma conta válida no computador para que o sistema possa controlar as políticas de segurança, mediante autenticação o usuário pode utilizar o sistema local e remotamente.

As características de acesso à usuários no Linux são rígidas, tornando-o um sistema extremamente seguro, veremos abaixo que existem dois tipos de usuário; os usuários normais e o usuário root (administrador).

7.1.1. Usuário root

Esta seção foi retirada do Manual de Instalação da Debian.

A conta root é também chamada de *super usuário*, este é um login que não possui restrições de segurança. A conta root somente deve ser usada para fazer a administração do sistema, e usada o menor tempo possível.

Qualquer senha que criar deverá conter de 6 a 8 caracteres, e também poderá conter letras maiúsculas e minúsculas, e também caracteres de pontuação. Tenha um cuidado especial quando escolher sua senha root, porque ela é a conta mais poderosa. Evite palavras de dicionário ou o uso de qualquer outros dados pessoais que podem ser adivinhados.

Se qualquer um lhe pedir senha root, seja extremamente cuidadoso. Você normalmente nunca deve distribuir sua conta root, a não ser que esteja administrando um computador com mais de um administrador do sistema.

Utilize uma conta de usuário normal ao invés da conta root para operar seu sistema. Porque não usar a conta root? Bem, uma razão para evitar usar privilégios root é por causa da facilidade de se cometer danos irreparáveis como root. Outra razão é que você pode ser enganado e rodar um programa *Cavalo de Tróia* -- que é um programa que obtém poderes do *super usuário* para comprometer a segurança do seu sistema sem que você saiba.

7.1.2. Usuário normal

Os usuários normais possuem permissão de escrita somente em seu diretório **home**, possui também acesso de execução nos diretórios **/bin/**, **/usr/bin/** e **/usr/local/bin/** (isto pode variar de distribuição para distribuição). Em alguns diretórios não possui permissão de leitura como o **/root/** e nas homes de outros usuários, estas permissões podem ser redefinidas, mas note bem que quanto menos poderes um usuário tiver mais seguro será o sistema.

As contas aqui também possuem senhas portanto os cuidados são os mesmos do usuário root, a única diferença é que se sua senha cair em mãos erradas o estrago será menor.

As contas de usuários no linux são bem flexíveis, por exemplo um usuário que acessa o servidor por uma máquina windows somente, não precisa de um shell válido no servidor Linux, vamos aprender a customizar este tipo de opção neste capítulo.

7.1.3. Criando um usuário

Vamos criar um usuário através do comando `useradd`, o comando `useradd` permite que especifiquemos de uma vez diversas opções sobre o usuário as mais comuns são `-g` , `-s` , `-G`, mas na tabela abaixo vou especificar mais opções deste comando:

| | |
|-----------------|--|
| <code>-u</code> | UID (número de identificação do usuário, caso seja omitido o sistema insere um número automaticamente) |
| <code>-g</code> | Grupo primário a que o usuário pertence (caso seja omitido o sistema cria um grupo com o nome do usuário) |
| <code>-G</code> | Grupos secundários do usuário (um usuário pode fazer parte de outros grupos) |
| <code>-d</code> | Diretório home do usuário (caso seja omitido o sistema vai criar um diretório /home/nomedousuário) |
| <code>-s</code> | Shell padrão do usuário (caso seja omitido o sistema vai selecionar o bash como padrão) |

Consultando as páginas do manual do comando `useradd` poderemos achar mais opções.

Vamos criar um usuário, veja o exemplo abaixo:

```
# useradd -g users -G audio bruno
```

Ok, usuário criado, automaticamente foi adicionado no grupo `users` e no grupo `audio`. Agora imagine como seria a criação de um usuário que acessa o servidor de uma workstation windows, o usuário não precisa de um shell válido. Para isso utilizamos um shell inválido chamado `"/bin/false"` veja abaixo sua criação:

```
# useradd -g users -s /bin/false bruno
```

7.1.4. Modificando um usuário

Da mesma maneira que criamos um usuário podemos modificar um usuário já criado com a ferramenta `usermod`, que tem uma sintaxe muito parecida com a `useradd`, na verdade as opções são as mesmas, a única novidade é a opção `-l` que permite a troca do nome de usuário.

Os usuários comuns não podem modificar as opções de suas próprias contas, o administrador deve executar estas alterações.

```
# usermod -l brunao bruno
```


7.1.5. Modificando o password

Até agora nada foi mencionado à respeito do password (senha) do usuário, a resposta é que os passwords vêm bloqueados por padrão nas distribuições atuais. O comando para criação e alteração de senhas de usuários e grupos é `passwd`, veja um exemplo de sua utilização:

```
# passwd bruno
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Por senhas serem secretas cada usuário pode alterar sua senha quando bem entender com o comando `passwd` sem nenhum parâmetro adicional, o administrador pode mudar a senha de um usuário, mas não pode saber qual era a senha anterior.

7.1.4. Excluindo um usuário

Para remover definitivamente usuários do sistema utilizamos a ferramenta `userdel`, adicionando a opção `-r` o sistema deleta todo o conteúdo do diretório home do usuário:

```
# userdel bruno
```



7.2 Grupos

Grupos servem para agrupar diversos usuários em um único grupo, isto é um tanto quanto óbvio. Exemplificando, temos um departamento pessoal na empresa, estes usuários podem acessar um diretório chamado `/usr/share/dpessoal`, e mais ninguém da empresa pode acessar esta pasta, o que fazemos para restringir o acesso? Ao invés de modificar a permissão de cada usuário que acessa este diretório, podemos criar o grupo **deptopessoal**, colocar os funcionários do departamento pessoal neste grupo e restringir o acesso ao diretório `/usr/share/dpessoal` para que somente quem faça parte deste grupo possa acessar.

7.2.1. Criando um grupo

Para criarmos um grupo utilizamos o comando `groupadd`, adicionando a opção `-g` podemos definir o **gid** (número de identificação do grupo) mas se for omitido o sistema vai adicionar a numeração automaticamente:

```
# groupadd deptopessoal
```


7.2.2. Excluindo um grupo

Para excluir um grupo utilize o comando `groupdel`, uma observação importante é que não podemos excluir os grupos primários de usuários, somente grupos secundários.

```
# groupdel deptopessoal
```



7.3 Permissões de acesso

No GNU/Linux todos os arquivos e diretórios são automaticamente assinalados por um usuário e um grupo que são "donos" deste arquivo/diretório, assim o sistema pode definir diferentes permissões de acesso entre usuários. Isto é muito importante pois um sistema multi usuário precisa ter uma camada de segurança especial, imagine que o usuário1 cria um diretório com documentos de texto importantes, o usuário2 por sua vez não sabe para que serve esta pasta e acaba apagando-a por engano, este tipo de coisa não pode acontecer em uma empresa. Desta forma foram criadas as permissões de acesso, que restringem as ações que dos usuários em arquivos e diretórios.

Vamos coletar dados sobre nossa conta atual para gerenciarmos corretamente as permissões de acesso.

7.3.1. Verificação de conta

Para saber que usuário estamos utilizando usamos o comando `whoami` na verdade pode parecer sem sentido saber que usuário estamos utilizando quando logamos, mas imagine que estemos acessando o servidor de outra maneira ou seja um usuário genérico.

```
$ whoami  
bruno
```

Precisamos saber também a que grupo pertencemos, utilizamos o comando `groups`:

```
$ groups  
users audio webdesign
```

Primeiro vem o grupo primário do usuário, logo após os grupos secundários. O comando `groups` pode retornar os grupos de outros usuários caso seja colocado seu username como parâmetro:

```
$ groups joao  
users audio video
```

Vamos agora verificar permissões de arquivos.

7.3.2. Verificação de permissões com `ls -l`

Para verificarmos as permissões de acesso à arquivos utilizamos o comando `ls -l`, que retorna uma lista com diversas colunas, devemos ter uma atenção especial com três destas colunas, Veja abaixo o exemplo:

```
$ ls -l /aulas/aula7/
-rw-r--r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

Veja na tabela abaixo que colunas devem ser observadas:

| | |
|------------|--|
| -rw-r--r-- | Permissões de acesso a este arquivo/diretório. |
| bruno | Usuário dono deste arquivo/diretório. |
| users | Grupo dono deste arquivo/diretório. |

Como podemos observar, as permissões de acesso são definidas por um conjunto de caracteres, no caso do **arquivo1.txt** as permissões de acesso são "-rw-r--r--", neste conjunto de caracteres estão definidas respectivamente as permissões de acesso do usuário dono, do grupo dono e dos outros usuários. Veja na figura abaixo como decifrar este triplet:



Agora podemos ver claramente, os caracteres em azul na figura definem as permissões de acesso ao dono do arquivo os caracteres em vermelho na figura definem as permissões de acesso ao grupo dono do arquivo e os caracteres em verde definem as permissões de acesso aos outros usuários. Podemos observar também que sempre são utilizados um conjunto de três caracteres para cada permissão. Vamos agora traduzir o que os caracteres representam:

| | |
|---|---|
| r | Leitura, simbolizada pela letra r (de <u>r</u> ead) |
| w | Escrita, simbolizada pela letra w (de <u>w</u> rite) |
| x | Execução, simbolizada pela letra x (de <u>e</u> xecute) |

Os caracteres sempre aparecem nesta ordem (rwx), caso o usuário não tenha

permissão de execução um sinal “ - ” vai aparecer no lugar do x (ficando rw-). Para tornar isto mais claro vamos observar o output do comando que executamos anteriormente:

```
-rw-r--r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

- O usuário dono possui permissão de leitura e escrita e não possui permissão de execução.
- O grupo dono possui permissão de leitura e não possui permissão de escrita nem de execução.
- Os outros usuários possuem permissão de leitura e não possuem permissão de escrita nem de execução.

7.3.3. Definindo posse de arquivos e diretórios por usuário com `chown`

Agora que já sabemos como verificar as permissões de acesso de arquivos e diretórios vamos aprender a definir estas permissões de acesso, primeiro vamos mudar o usuário dono deste arquivo, para isso utilizamos o comando `chown`, sua utilização é bem simples veja abaixo um exemplo:

```
# chown root /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rw-r--r-- 1 root users 0 May 27 08:14 arquivo1.txt
```

O usuário dono foi alterado, o comando `chown` pode também alterar de uma única vez o usuário e o grupo, passando como parâmetro usuário:grupo, veja o exemplo abaixo:

```
# chown bruno:webdesign /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rw-r--r-- 1 bruno webdesign 0 May 27 08:14 arquivo1.txt
```

7.3.4. Definindo posse de arquivos e diretórios por grupo com `chgrp`

Existe também um comando que altera somente o grupo dono de determinado arquivo/diretório, este comando é o `chgrp` sua utilização é igualmente simples:

```
# chgrp users /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rw-r--r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

7.3.5. O comando chmod

Enquanto os comando `chown` e `chgrp` servem para alterar respectivamente o usuário dono e o grupo dono, o comando `chmod` serve para alterar as permissões `rwX` vistas anteriormente, o comando `chmod` recebe dois argumentos: a nova permissão e o objeto de destino. Você poderá alterar as permissões de acesso a um objeto de duas maneiras, a primeira é através dos triplets (`rwX`) vistos anteriormente, a segunda é através de uma tabela numérica que veremos adiante.

7.3.6. Definindo permissões de acesso por triplets

Primeiro, vamos definir esta permissão por triplets. Veja o exemplo abaixo:

```
# chmod g+w /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rw-rw-r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

Como podemos observar foi utilizado o parâmetro `g+w`, afinal o que isto significa? Quer dizer que o grupo pode escrever no **arquivo1.txt**. Então fica claro que os parâmetros passados são o tipo de **usuário + permissão**. Veja na tabela abaixo que tipos de usuários podem ser utilizados:

| | |
|---|-------------------|
| u | Usuário dono |
| g | Grupo dono |
| o | Outros usuários |
| a | Todos os usuários |

Vamos a mais um exemplo para que fique mais claro:

```
# chmod u+x /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rwxrw-r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

Como podemos observar foi utilizado o parâmetro `u+x`, isto significa que o usuário dono pode executar o **arquivo1.txt**.

Como fazemos para retirar permissões? Utilizamos o símbolo “ - ” ao invés do símbolo “ + ” Veja o exemplo abaixo:

```
# chmod u-x /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rw-rw-r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

No lugar do `rwX` encontramos o `rw-` isto significa que retiramos a permissão de execução do usuário dono.

7.3.7. Definindo permissões de acesso pelo modo numérico

A definição de permissões pelo modo numérico funciona de maneira diferente; de uma única vez definimos as permissões do usuário dono, do grupo dono e dos outros usuários através de um número passado como parâmetro, a tabela de números esta abaixo:

| Número | Triplet |
|--------|---------|
| 7 | rwX |
| 6 | rw- |
| 5 | r-X |
| 4 | r-- |
| 3 | -wX |
| 2 | -w- |
| 1 | --X |
| 0 | --- |

Veja o exemplo abaixo a utilização do modo numérico:

```
# chmod 664 /aulas/aula7/arquivo1.txt
$ ls /aulas/aula7/
-rw-rw-r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

Utilizamos o parâmetro 664; o primeiro número serve para o usuário dono, o segundo para o grupo dono e o terceiro para os outros usuários.



7.4 Permissões especiais

Apesar do esquema de permissões do Linux ser bem completo, existem outros tipos de permissões chamadas de permissões especiais, estas permissões são utilizadas em casos especiais onde as permissões padrão não funcionam. por exemplo a execução de um programa de administração do sistema por um usuário qualquer ou então criação de um diretório com permissão de escrita para todos.

7.4.1. Umask

Quando um novo arquivo é criado, podemos observar que suas permissões são por padrão -rw-r--r--, isto acontece porque todos os arquivos criados com a permissão 666 (leitura e escrita permitida para todos, o que é um tanto quanto inseguro) então o Linux consulta uma variável do sistema chamada umask que retira a permissão de escrita do grupo e dos outros usuários. Consulte a variável umask digitando o comando `umask` no console:

```
$ umask
0022
```

Olhando novamente na tabela podemos constatar que o número 0 significa ---, e o número 2 significa -w-, ou seja a umask vai retirar o -w- do grupo e dos outros. Veja o exemplo abaixo a alteração da variável umask para que os outros usuários não possam ler os arquivos quando eles são criados:

```
$ umask 0027
```

7.4.2. Suid

Todos os processos iniciados pelo usuário possuem as permissões de acesso deste usuário, isto ocorre pois um processo quando é executado por um usuário utiliza seu Suid. Diante disto qualquer programa iniciado por nós não pode acessar arquivos e diretórios os quais não tenhamos acesso, esta é uma das maiores chaves de segurança do Unix. Vamos ao exemplo, o arquivo de passwords do sistema, /etc/passwd, não pode ser alterado diretamente pelos usuários, pois nas suas permissões podemos constatar que permite que os outros usuários o subescrevam:

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 1451 May 25 16:29 /etc/passwd
```

Então de que maneira mudamos nossa senha? Vimos anteriormente que é através do programa passwd, e o programa passwd possui uma permissão especial chamada suid, ou seja todas as ações executadas por este programa rodam com o suid do usuário dono (no caso o root) e não com as permissões do usuário que o executa, com as permissões de usuário root, o programa pode tranquilamente alterar o arquivo /etc/passwd.

Vejamos abaixo como identificar quando um programa possui suid:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 26616 Dec 23 19:40 /usr/bin/passwd
```

No primeiro triplet de usuário podemos observar que ao invés do caractere x encontramos o caractere s, isto indica que aquele programa possui a permissão suid. Para alterarmos a permissão suid utilizamos o parâmetro u+s:

```
$ chmod u+s /aulas/aula7/arquivo1.txt
-rwSr-w-r-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

Uma informação importante que podemos constatar agora, quando o comando possui o flag x (de execução) ligado o caractere s aparecerá minúsculo, se o comando não possuir o flag x o caractere S aparecerá maiúsculo.

7.4.3. Sgid

A permissão sgid funciona da mesma maneira que o suid, ou seja, permite que programas utilizem a permissão do grupo ao invés da permissão de quem utiliza. Sua utilização é semelhante à utilização do comando suid, veja um exemplo:

```
$ chmod g+s /aulas/aula7/arquivo1.txt  
-rwSrwsr-- 1 bruno users 0 May 27 08:14 arquivo1.txt
```

Sua utilização é bastante útil em diretórios compartilhados, por exemplo quando um diretório possui o sgid ligado, todos os objetos criados dentro deste diretório irão pertencer ao grupo dono deste diretório.

7.4.3. Sticky

Trabalhar com diretórios compartilhados é perigoso, a partir do momento que todos os usuários tem acesso à todos os arquivos, eles podem simplesmente deletar qualquer arquivo, inclusive os arquivos de outros usuários, para prevenir estes acontecimentos o Linux possui um outro modelo de permissão chamado sticky, que permite que um arquivo ou diretório só possa ser apagado pelo usuário dono ou pelo root. Um exemplo do sticky é o diretório /tmp:

```
$ ls -ld /tmp  
drwxrwxrwt 12 root root 4096 May 27 14:03 /tmp/
```

No último triplet podemos observar o caractere t no lugar do x, o que indica que o flag stick está ligado. Para adicionar a permissão sticky a um arquivo ou diretório utilize o comando `chmod +t arquivo`.



Exercícios

1. Crie um usuário normal com o shell /bin/false, sua home /dev/null, desbloqueie seu password colocando uma senha qualquer, depois disto tente logar em um outro terminal (ALT+F2) com este usuário, observe atentamente o que ocorre.
2. Crie um grupo admin e adicione o usuário recém criado a este grupo.
3. Remova o usuário e o grupo criados anteriormente.
4. Crie um arquivo e altere suas permissões para que o grupo e os outros usuários possam escrever neste arquivo.
5. Modifique com um único comando a permissão do arquivo recém criado para que a mesma fique assim: "rwxrw-r--"



Vamos agora iniciar a instalação de novos programas no GNU/Linux, vamos aprender como compilar e instalar programas pelo código fonte, também vamos aprender como trabalhar com pacotes RPM e DEB, utilizados em diversas distribuições. Finalmente iremos gerenciar programas instalados, atualizar estes programas e desinstala-los quando for necessário. Ao final deste capítulo você será capaz de:

- Compilar e instalar e desinstalar programas pelo código fonte.
- Resolver dependências de compilação.
- Instalar, atualizar e desinstalar pacotes RPM.
- Instalar, atualizar e desinstalar pacotes DEB via apt-get.
- Instalar, atualizar e desinstalar pacotes DEB.
- Resolver dependências binárias.



8.1. Instalando novos programas

A instalação de programas no Linux pode ser feita de diversas maneiras, e oferecendo diversas opções e customizações. Grande parte dos programas disponíveis está em código fonte, o que permite que alteremos o código se precisarmos (visando melhorias de hardware por exemplo), estes programas são compilados em nossa máquina, logo após são instalados. Podemos também encontrar pacotes pré-compilados chamados de pacotes binários, este pacotes contém o programa pré-compilado genericamente, possibilitando a instalação imediata.

O processo de instalação em si não é difícil como pensam os iniciantes, é até bem fácil, em pouco tempo torna-se praticamente automático. Sua única dificuldade (que ocorre de vez em quando) é a resolução de dependências, estas dependências são geralmente bibliotecas ou outros programas que precisam estar instalados para que possamos continuar a instalação do programa atual.

Para o usuário final, trabalhar com pacotes binários é a maneira mais fácil de gerenciar programas, já o processo de compilação tem como público alvo desenvolvedores e usuários avançados pois permitem que especifiquemos opções na hora da compilação, à caráter didático ambos os processos serão cobertos neste capítulo.



8.2. Compilando pelo código-fonte

Na sua convivência com Linux vai perceber que existe toda uma comunidade de desenvolvedores de software open-source, ou seja, os programadores desenvolvem programas e não cobram nada pela sua utilização, inclusive liberam o código fonte. Desta forma vamos encontrar quase que a totalidade de programas no Linux disponíveis em código fonte (existem algumas exceções, por exemplo Skype, Adobe Acrobat e etc.).

8.2.1. Obtendo programas

Existe uma grande massa de software livre on-line, para gerenciar toda essa massa de dois grandes portais gerenciam projetos de software livre:

- <http://sourceforge.net>
- <http://freshmeat.net>

Além de gerenciar, estes portais hospedam páginas do projeto e também os arquivos de código fonte e binários. Existe também uma lista de softwares bem completa no Fórum Guia do Hardware, site <http://forumgdh.net/viewtopic.php?t=50420>, esta lista é bem útil quando não se sabe ao certo que programa vai atender à sua necessidade.

Vamos iniciar a instalação de um programa, como exemplo vou instalar o programa Links é um navegador de internet no modo texto, primeiro vou procurar o programa no Google: <http://www.google.com.br/linux/> , logo de cara já achamos o link para o freshmeat.net, ao clicarmos encontramos diversas informações como a página oficial, o estado de desenvolvimento e também os arquivos para download, recomendo que leia atentamente estas informações pois nas empresas, computadores e servidores de trabalho não podem ser utilizados como ambiente de testes, procure sempre a versão mais estável o possível, leia também a documentação do software, quanto mais

informação puder ser agregada, menor é a probabilidade de erros e problemas. Vamos agora obter o programa, faça o download do arquivo. Para um maior controle do que é instalado recomendo que mantenha seus sources em um único diretório, **/usr/src** pode ser utilizado como padrão. Com o arquivo salvo neste diretório vamos iniciar o processo de instalação.

8.2.2. Descompactando/Desempacotando

Quando obtemos programas pelo código fonte sua extensão é geralmente **.tar.gz** ou **.tar.bz2**, estes nada mais são do que um conjunto de arquivos compactados, descompactamos os mesmos com a ferramenta **tar**, as opções diferem de um para o outro, quando temos um pacote **.tar.gz** utilizamos o comando **tar -zxvf nomedopacote.tar.gz**, quando temos um pacote **.tar.bz2** utilizamos o comando **tar -jxvf nomedopacote.tar.bz2**. Depois deste comando será criado um novo diretório com o nome do programa, veja o exemplo abaixo:

```
$ ls
links-2.1pre17.tar.gz
$ tar -zxvf links-2.1pre17.tar.gz
```

A listagem é grande devido à opção **v**, portanto será omitida aqui. Pronto, temos um pacote descompactado. Vamos agora analisar arquivos importantes antes da instalação.

8.2.3. Arquivos README/INSTALL

Estes arquivos são muito importantes, o arquivo **README** vai conter todas as informações sobre o programa, desenvolvedores, licença, site onde encontrar, perguntas frequentes e muitas outras informações importantes, recomendo que leia atentamente este arquivo pois se tiver alguma dúvida existe grande probabilidade de encontrar a resposta aqui. Já o arquivo **INSTALL** cobre as informações para a instalação dos programas bem como as customizações para os desenvolvedores, deve ser consultado eventualmente caso precise de ajuda na instalação ou queira ativar uma característica especial do programa na hora da compilação.

8.2.4. Script configure

O script **configure** é um script criado pelos desenvolvedores do programa utilizando a ferramenta **automake**, na verdade sua principal função é gerar um arquivo chamado **Makefile**, que vai conter as informações da sua máquina para a compilação do programa, estas informações são as dependências e localização das mesmas, este script é útil pois as dependências podem variar de máquina para máquina e distribuição para distribuição, o que torna cada sistema diferente fazendo-se necessária uma ferramenta como esta. Alguns programas muito antigos não tem este script, fazendo necessária a alteração manual de seu **Makefile**. Para iniciar o script simplesmente o execute com **./:**

```
$ ./configure  
checking for ...
```

A listagem é bem grande, geralmente as dependências serão listadas aqui, caso ocorra alguma pule para a seção 8.2.8, esta é a hora em que podemos passar opções para a instalação do programa, a opção `--prefix=/diretório` por exemplo instala no diretório que você escolher, ou no caso do links `--enable-graphics` vai habilitar o modo gráfico, (li isto no arquivo INSTALL), veja o exemplo abaixo de como executar o script `configure` com opções:

```
$ ./configure --enable-graphics  
checking for ...
```

Agora que arquivo Makefile foi gerado podemos passar para a próxima fase, a compilação.

8.2.5. Compilando com make

A ferramenta `make` serve para gerenciar a compilação de diversos arquivos e que compiladores utilizar para estes arquivos. Sua utilização é bem simples, digite `make` na linha de comando e espere alguns instantes (isto varia muito do tamanho do programa, da capacidade da máquina, pode demorar de segundos à minutos ou até horas em programas muito volumosos).

```
$ make  
...
```

Com isto os programas serão compilados, vamos agora ao passo final da instalação.

8.2.6. Instalando com make install

O passo final é simplesmente rodar como root o comando `make install` (como root pois os usuários normais não tem acesso ao diretório `/usr/local` e `/usr/bin` onde geralmente os arquivos são instalados). Veja abaixo sua utilização:

```
# make install  
...
```

Programa instalado. Pode parecer complicado à princípio mas você verá no resumo adiante o quão fácil é a instalação.

8.2.7. Resumo da instalação por código fonte

Vou agora resumir toda esta seção em três linhas de comandos:

```
$ ./configure  
$ make  
# make install
```

Caso não haja nenhuma dependência, 98% dos programas que você tiver que instalar pelo código fonte serão as linhas acima.

8.2.8. Resolvendo problemas

De vez em quando no processo de instalação por código fonte, não conseguimos instalar programas imediatamente pois ou o script `configure` ou a ferramenta `make` retornam algum erro, portanto devemos resolver este erro para o sucesso da instalação.

Na maioria das vezes sua causa é a falta de alguma biblioteca, uma vez que o script `configure` ou a ferramenta `make` sintam falta de alguma biblioteca necessária para a compilação do programa, o processo será imediatamente abortado e você terá que procurar esta biblioteca, caso ela esteja em sua máquina você deverá passar o seu endereço como parâmetro para o script `configure` (isto é muito difícil acontecer), a outra opção é que seu sistema simplesmente não possua a biblioteca necessária, o que significa que você precisa procurar na internet e instalar esta biblioteca. As bibliotecas necessárias estão explícitas nos arquivos `README` e `INSTALL`.

Outras vezes a instalação não funciona por motivos desconhecidos, que podem variar de instalação para instalação, à medida que seu conhecimento na utilização do Linux aumenta, fica mais fácil a resolução destes problemas, volto a recomendar; leia o arquivo `INSTALL` sempre que tiver algum problema, busque no Google como fixar este problema e pergunte em fóruns na web.

8.2.9. Desinstalando com `make uninstall`

Caso o programa já esteja instalado corretamente, sua desinstalação é semelhante à instalação, vá ao diretório `source` e digite `make uninstall`, aqui é que vai a importância de manter uma árvore de `sources` bem organizada.

```
# make uninstall
```

Programa desinstalado.



8.3. Trabalhando com pacotes binários

No início do capítulo foi mencionado que existem pacotes binários, afinal o que são? São propriamente pacotes de programas pré-compilados que já vem prontos para a instalação, todas as distribuições de Linux utilizam alguma forma de empacotamento. Pacotes foram um grande passo na distribuição do Linux, acabando com problemas de falta de bibliotecas e criando um gerenciamento inteligente de dependências.

A instalação de pacotes binários oferece algumas vantagens sobre a instalação à partir do código fonte:

- Fácil instalação, atualização e desinstalação.
- Proteção de arquivos
- Gerenciamento inteligente de dependências.
- Fácil gerenciamento de programas instalados.

Pacotes binários são de fato uma solução para distribuição de programas, porém, devemos analisar suas desvantagens:

- Programas compilados pelo código fonte possuem melhor performance.
- Pacotes de distribuições diferentes seguem padrões diferentes.
- Pacotes possuem dependências de outros pacotes.
- A criação de pacotes é complicada e requer conhecimentos avançados.
- Caso haja a danificação do banco de dados dos pacotes, o sistema vai tornar-se instável.

Trabalhar com pacotes pode ser muito útil, uma mão-na-rodinha em determinados casos, mas requer cuidados. Caso o usuário prefira instalar por pacotes, deve seguir sempre as seguintes recomendações:

1. Nunca instale pacotes de outras distribuições, cada distribuição utiliza seu padrão, de diretórios bibliotecas e arquivos, isto vai tornar seu sistema instável.
2. Nunca force a instalação de pacotes ou ultrapasse as dependências de pacotes à menos que saiba exatamente o que está fazendo.
3. Leia os manuais do rpm ou deb para saber manusear corretamente pacotes.



8.4. Empacotamento RPM

Sistema de empacotamento RPM (**R**ed **H**at **P**ackage **M**anager), foi desenvolvido pela empresa Red Hat, criadora da distribuição Red Hat Linux. É um sistema de pacotes robusto que tornou-se padrão não só na distribuição Red Hat, mas em diversas outras distribuições como Suse, Mandrake, Conectiva, Yellow Dog, ArchLinux, entre outras. O controle de pacotes RPM é feito através do console mas existem diversas interfaces gráficas e também interfaces on-line que permitem o gerenciamento de pacotes RPM.

8.4.1. Obtendo pacotes RPM

Geralmente distribuições baseadas no empacotamento RPM possuem em seus sites um repositório com pacotes binários RPM. Vamos tomar como exemplo a distribuição Fedora navegando pelo site oficial encontramos geralmente na seção downloads o repositório com os pacotes de todas as versões, exemplo: <http://download.fedora.redhat.com/pub/fedora/linux/core/3/i386/os/Fedora/RPMS/>.

Para outras distribuições esta regra é a mesma, procure sempre no site oficial os repositórios de pacotes. Um outro método é procurar em ferramentas de buscas especiais para pacotes RPM:

- <http://www.rpmfind.net>
- <http://rpmseek.com>

Estas ferramentas nada mais são do que links para os repositórios oficiais. Em último caso procure no Google o pacote do programa, mas lembre-se de que não estando nos repositórios oficiais o pacote pode não funcionar e talvez até danificar o banco de dados RPM, se não encontrar o pacote prefira compilar pelo código fonte.

8.4.2. Instalando pacotes RPM

Para vias de exemplo será utilizado o pacote elinks-0.9.1-1.i386 da distribuição Fedora Core 2. Com o pacote em mãos instalamos utilizando o comando `rpm -i nomedopacote.rpm`, caso queira exibir o progresso da instalação recomendo que utilize também a opção `-vh` veja o exemplo abaixo:

```
# rpm -ivh elinks-0.9.1-1.i386.rpm
warning: elinks-0.9.1-1.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
Preparing... ##### [100%]
 1:elinks ##### [100%]
```

Programa instalado, a opção `-v` serve para que o processo seja comentado na tela, a opção `-h` serve para exibir a barra de progresso formada pelos `###`.

8.4.3. Resolvendo dependências

Normalmente, pacotes binários contém dependências, um exemplo, você não pode instalar o programa xpdf sem ter instalado antes as bibliotecas do X (servidor gráfico).

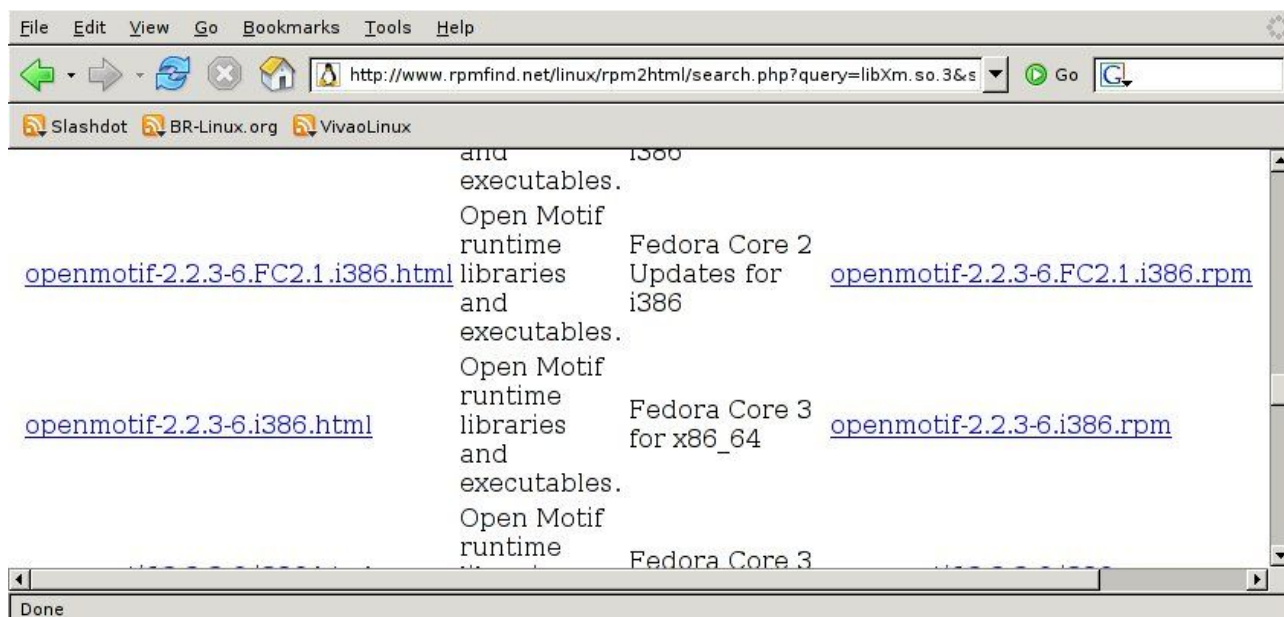
Um dos pontos fortes dos pacotes binários é o gerenciamento de dependências. Caso haja alguma dependência no momento da instalação o processo é abortado e o prompt imediatamente relata que pacotes ou bibliotecas que devem ser instalados antes que se instale o pacote atual. Para verificar quais são as dependências de um pacote antes de instalar digite o comando `rpm -qpR nomedopacote`, veja o exemplo abaixo:


```
# rpm -qpR xpdf-3.00-3.i386.rpm
warning: xpdf-3.00-3.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
...
...
```

Utilizando as ferramentas de busca que você viu no capítulo 5 poderia procurar por essas bibliotecas em seu pc, caso não encontre procure na web pelo nome da dependência, veja o exemplo abaixo

```
# rpm -ivh xpdf-3.00-3.i386.rpm
warning: xpdf-3.00-3.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
error: Failed dependencies:
        libXm.so.3 is needed by xpdf-3.00-3
```

O pacote não pode ser instalado pois a biblioteca libXm.so.3 está faltando.Ok Vamos procurar na web por esta biblioteca, utilizando o site rpmfind.net, procure pela biblioteca libXm.so.3, depois de obter a resposta procure a sua distribuição correta:



Achamos o pacote openmotif-2.2.3-6.FC2.1.i386.rpm, precisamos instalar este pacote para resolvermos a dependência do xpdf.

8.4.4. Atualizando pacotes

O processo de atualização é simples, utilize o comando `rpm -Uvh nomedopacote.rpm`, vamos como exemplo atualizar o pacote do elinks:

```
# rpm -Uvh elinks-0.9.2-2.i386.rpm
warning: elinks-0.9.2-2.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
Preparing... ##### [100%]
1:elinks ##### [100%]
```

8.4.5. Forçando a instalação de pacotes

Apesar do gerenciamento de dependências ajudar em diversas ocasiões, existem outras que caímos em um problema vamos exemplificar; tentamos instalar o pacote X.rpm, o pacote X depende do pacote Y, baixamos então o pacote Y e ao instalar ele acusa que depende do pacote X, e agora? Isto pode parecer brincadeira mas acontece, a solução é forçar a instalação de um pacote para depois instalar o outro. No caso de dependências somente utilizamos a opção `--nodeps`, no exemplo vou instalar o programa xpdf sem satisfazer a dependência:

```
#rpm -ivh --nodeps xpdf-3.00-3.i386.rpm
warning: xpdf-3.00-3.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
Preparing... ##### [100%]
 1:xpdf ##### [100%]
```

Observe que o programa esta instalado agora, mas não vai funcionar pois mesmo instalado ele continua dependendo do arquivo libXm.so.3 que está faltando. Utilize o `--nodeps` somente em casos semelhantes aos do exemplo X Y acima.

Existe uma outra maneira de forçar a instalação, vamos supor que você tenha apagado acidentalmente o arquivo de configuração de algum pacote que já está instalado, que tal reinstalar o pacote para corrigir o problema? Veja o exemplo abaixo:

```
# rpm -ivh elinks-0.9.2-2.i386.rpm
warning: elinks-0.9.2-2.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
Preparing... ##### [100%]
package elinks-0.9.2-2 is already installed
```

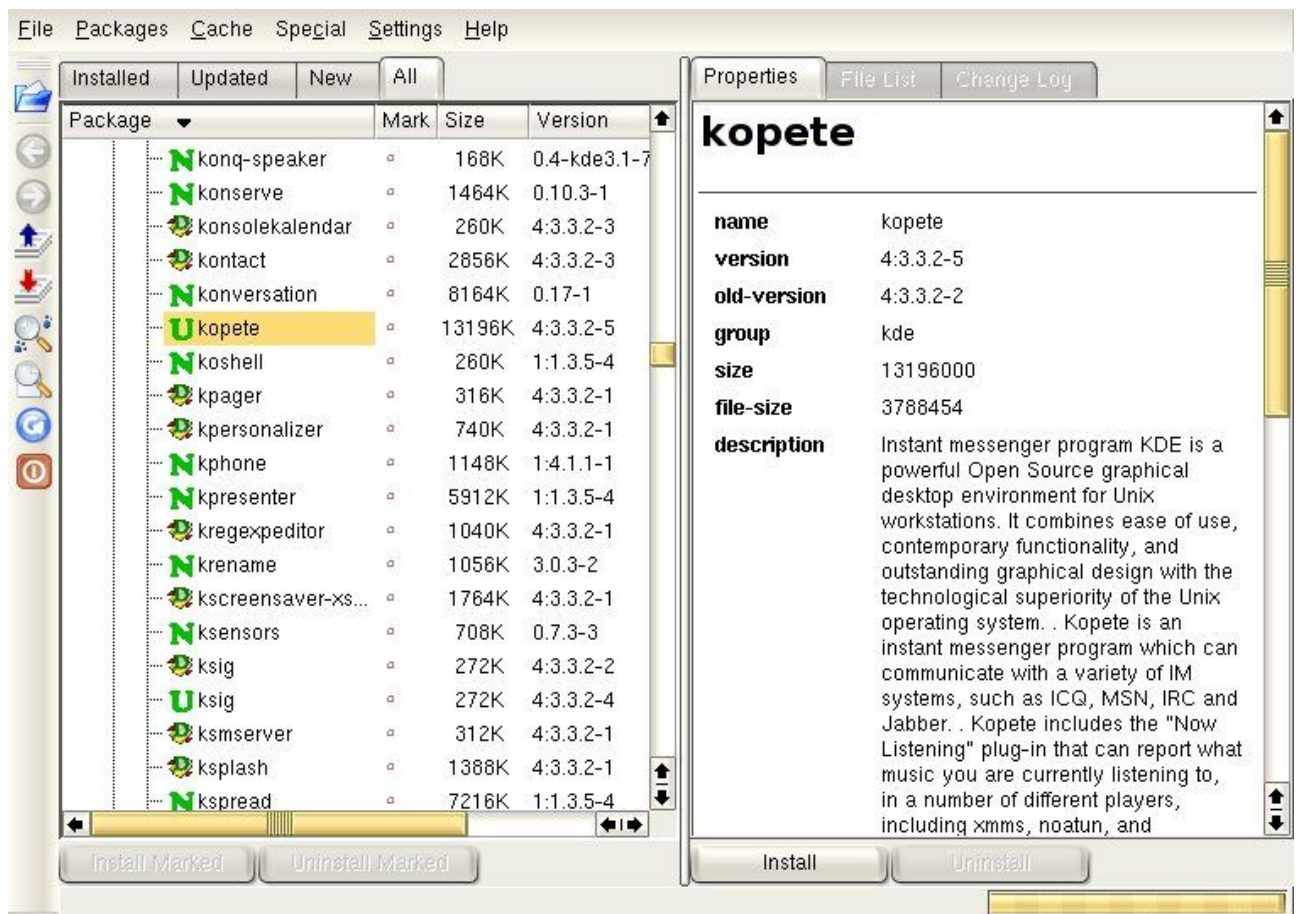
A instalação falhou, a solução neste caso seria forçar a instalação, utilizamos a opção `--force`:

```
# rpm -ivh --force elinks-0.9.2-2.i386.rpm
warning: elinks-0.9.2-2.i386.rpm: V3 DSA signature: NOKEY, key ID 4f2a6fd2
Preparing... ##### [100%]
 1:elinks ##### [100%]
```

Utilize com cuidado as opções `--nodeps` e `--force` pois elas podem bagunçar o sistema.

8.4.6. Gerenciamento de pacotes RPM

O gerenciamento de pacotes RPM visto até agora é totalmente textual, porém tudo pode ser feito através da interface gráfica, auxiliando os usuários. Existem diversas ferramentas que gerenciam pacotes, cada distribuição inclui sua ferramenta de gerenciamento, existem também ferramentas genéricas como o kpackage da suíte KDE, que permitem um gerenciamento completo de pacotes independente de distribuição, veja no exemplo abaixo o programa kpackage:



Existem vantagens na interface gráfica, é muito mais rápido gerenciar seus pacotes por estas ferramentas, através delas podemos ver os pacotes instalados, os pacotes disponíveis no servidor e também os pacotes que podem ser atualizados. O bom administrador deve saber como gerenciar de ambas as formas, a interface gráfica no entanto vai ser uma mão na roda muitas vezes. Abaixo vou colocar os links para outras ferramentas gráficas de gerenciamento de pacotes:

- Suse YaST: http://www.suse.net.au/en/private/products/suse_linux/i386/yast.html
- RedHat: <http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/custom-guide/ch-graphical-rpm.html>
- Mandrake: <http://www.mandrakelinux.com/en/demos/Spotlight/SoftwareMgr/>

8.4.7. Desinstalando pacotes RPM

A desinstalação de pacotes RPM pode ser feita pelos utilitários gráficos ou pelo console com a opção -e, as regras de dependência aqui também contam, se você desinstalar o pacote X e o pacote Y depender dele o pacote Y não mais vai funcionar. Vamos desinstalar o pacote xpdf.

```
# rpm -e xpdf
```

A boa notícia é que não precisamos escrever o nome completo dos pacotes.

Recomendo novamente que leia o manual do `rpm`, existe muita informação útil nestas páginas, e é um conhecimento universal, diversas distribuições utilizam o RPM como padrão e saber como trabalhar com isso vai ajudar na utilização destas distribuições.



8.5. Empacotamento Debian

Um outro padrão utilizado são os pacotes DEB, utilizados pela distribuição Debian e outras distribuições baseadas em Debian (Knoppix, Kurumin). Pacotes DEB são mantidos pela equipe Debian e passam por um controle rigoroso de testes. Todos os pacotes Debian oficiais possuem um mantenedor responsável por este pacote, o que resulta em um sistema muito bem organizado. O controle de segurança também é bem rígido quanto aos pacotes da distribuição, além do longo período de testes a que são submetidos os pacotes, existe um repositório somente com pacotes livres de vulnerabilidades.

8.5.1. Ferramenta `apt`

A distribuição Debian por padrão utiliza o sistema de gerenciamento de pacotes `apt` (Advanced Packaging Tool), a principal função desta ferramenta é a instalação e atualização de pacotes. Uma característica positiva do `apt` é que ele gerencia as dependências e as instala automaticamente, poupando o trabalho de procurar cada dependência.

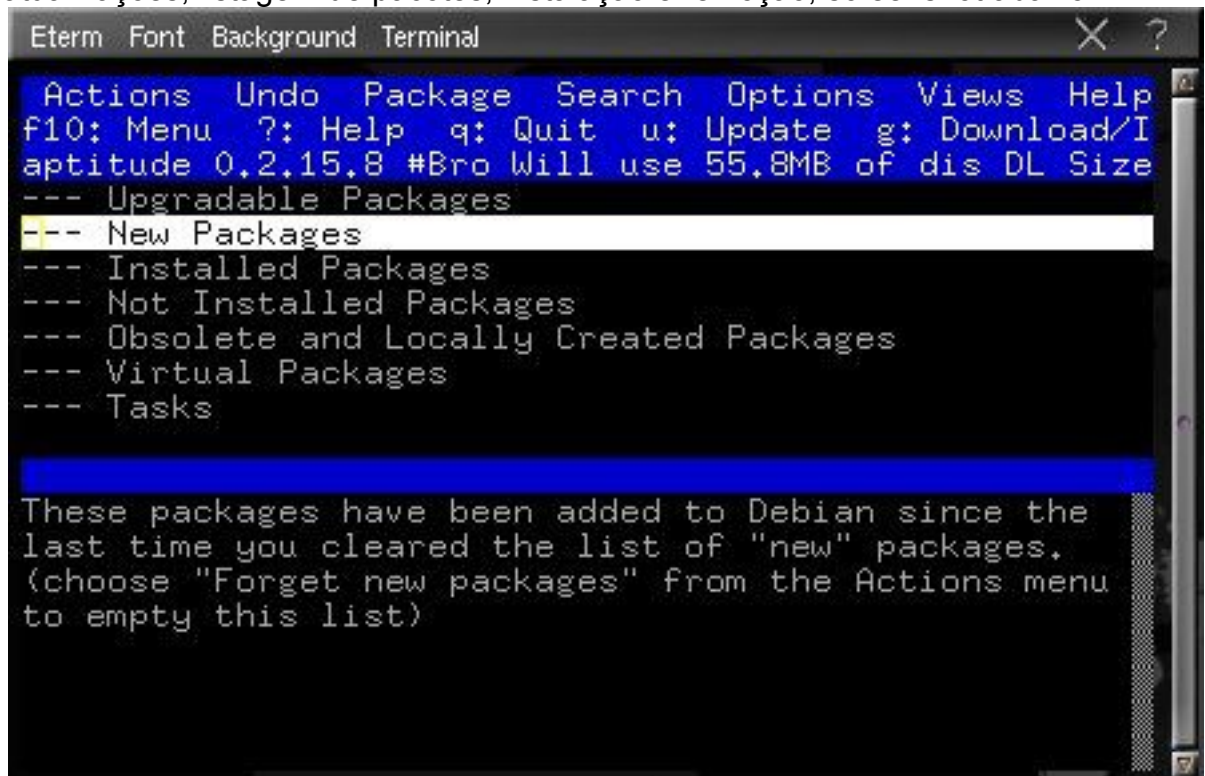
O `apt` no entanto é somente um gerenciador, para que funcione é necessário um frontend, o mais básico (e mais utilizado) é o `apt-get`, sua utilização consiste na seguinte sintaxe: `apt-get <ação> <pacote>`, abaixo uma tabela de ações:

| | |
|-------------------------------------|--|
| <code>install <pacote></code> | Instala novos pacotes. |
| <code>update</code> | Atualiza a lista de pacotes disponíveis obtida nos repositórios contidos no arquivo <code>/etc/apt/sources.list</code> |
| <code>upgrade</code> | Realiza a atualização dos pacotes de sua distribuição |
| <code>dist-upgrade</code> | Realiza a atualização de sua distribuição |
| <code>remove <pacote></code> | Remove pacotes instalados. |

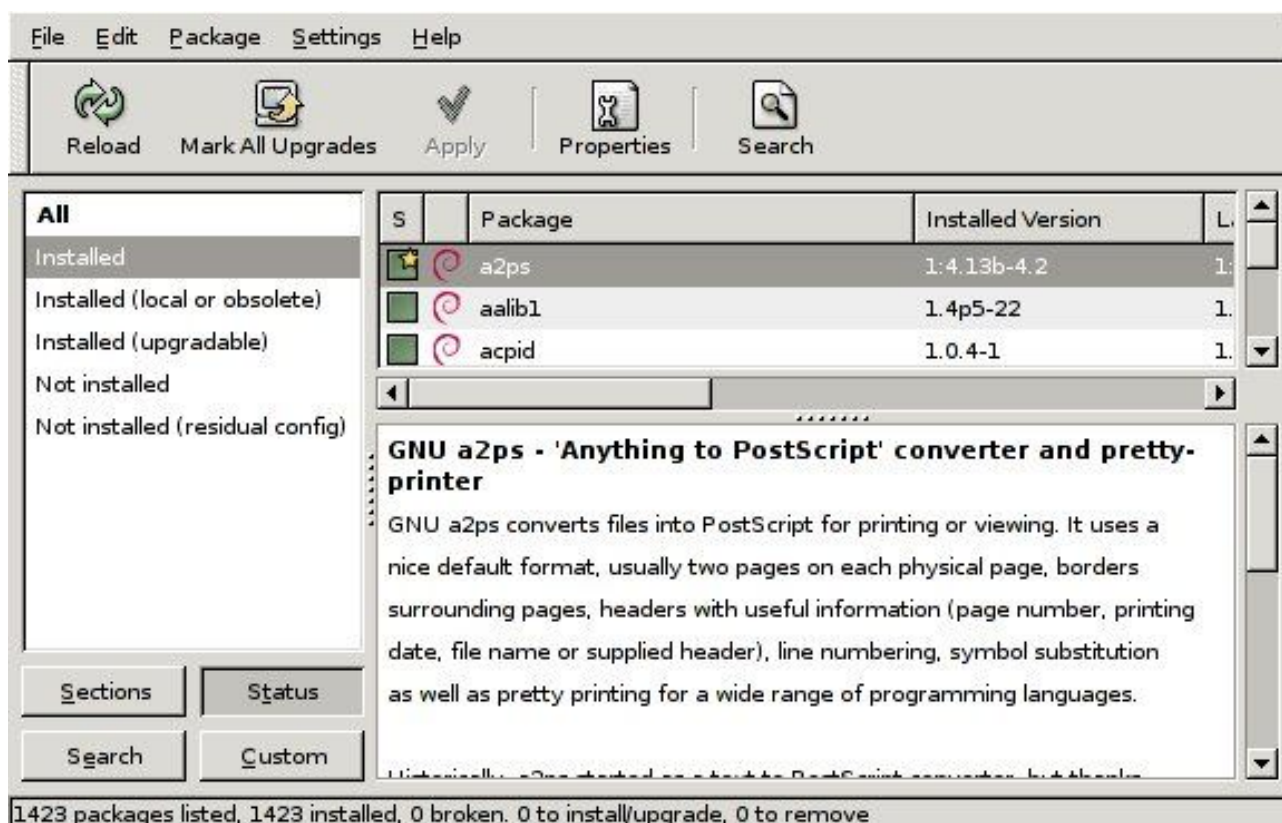
Existem dois outros frontends para o `apt` que merecem destaque:

- A `aptitude` é um frontend modo-texto, a utilização de comandos é a mesma do `apt-get` (`install`, `update`...), possui também uma interface no console para o gerenciamento de

atualizações, listagem de pacotes, instalação e remoção, screenshot abaixo:



- Synaptic é um front end gráfico, tem as mesmas características do aptitude com a vantagem de ser gráfico e permitir a navegação pelo mouse, screenshot:



Com a introdução já realizada vamos agora aprender a utilizar o apt.

8.5.2. O arquivo /etc/apt/sources.list

Este arquivo contém a lista de repositórios que serão utilizados como fonte de dados para downloads de pacotes, um exemplo de entrada do sources.list:

```
deb http://ftp.debian.org/debian/ distribuição secao1 secao2 secao3
deb-src http://ftp.debian.org/debian/ distribuição secao1 secao2 secao3
```

A tabela abaixo explica a formatação destas linhas:

| | |
|---|--|
| deb / deb-src | Tipo de pacote, pacote binario ou sources do pacote. |
| http://ftp.debian.org/debian/ | Endereço do repositório. |
| distribuição | Tipo de distribuição: <ul style="list-style-type: none"> ● stable ● testing ● unstable |
| seção | Seção a que pertence o pacote, veja relação em http://www.debian.org.br/doc/debian-policy/ch-archive.html#s-sections |

Abaixo um exemplo de sources.list real:

```
# cat /etc/apt/sources.list
deb http://ftp.br.debian.org/debian/ unstable main non-free contrib
deb-src http://ftp.br.debian.org/debian/ unstable main non-free contrib
deb http://non-us.debian.org/debian-non-US unstable/non-US main contrib
non-free
deb-src http://non-us.debian.org/debian-non-US unstable/non-US main
contrib non-free
deb http://ftp.us.debian.org/debian/ unstable main non-free contrib
```

Todos os repositórios oficiais Debian possuem o mesmo sincronismo de informações, isto significa por exemplo que o repositório do Brasil possui informações e pacotes tão atualizados quanto o repositório dos EUA.

Existe uma informação muito importante, são os repositórios de arquivos inscritos no arquivo `/etc/apt/sources.list` que vão definir se sua distribuição será *stable* (estável), *testing* (de testes), *unstable* (instável). Uma breve explicação (retirada de <http://www.debian.org.br/releases/>):

stable

A distribuição “stable” contém a última distribuição oficialmente lançada pela Debian.

Essa é a versão de produção do Debian, ela que é recomendada primeiramente.

A distribuição “stable” do Debian GNU/Linux está atualmente na versão 3.0r5 e seu codinome é *woody*. Ela foi lançada em 16 de Abril de 2005.

testing

A distribuição “testing” contém pacotes que não foram aceitos numa versão “stable” ainda, mas eles já estão na fila para serem aceitos. A principal vantagem de usar essa distribuição é que ela tem versões mais novas dos softwares. O suporte de segurança oficial para esta distribuição começou em 3 de maio de 2005.

Veja o [FAQ do Debian](#) para mais informações sobre [o que é “testing”](#) e [como ela se torna “stable”](#).

A distribuição “testing” atual chama-se *sarge*.

unstable

É na distribuição “unstable” que o desenvolvimento ininterrupto do Debian ocorre. Geralmente, os usuários dessa distribuição são os próprios desenvolvedores e pessoas que gostam de emoções fortes.

A distribuição “unstable” atual chama-se *sid*.

8.5.3. Atualizando a lista de pacotes disponíveis

Com os repositórios já indicados vamos agora atualizar a lista de pacotes do apt, digite o comando `apt-get update`:

```
# apt-get update
```

O download vai iniciar, pode demorar alguns minutos dependendo da sua conexão. Após o download a lista de pacotes vai estar atualizada, esta lista é a fonte de todas as outras ações portanto deve sempre que possível estar atualizada.

8.5.4. Instalando pacotes Debian via apt

A instalação de pacotes via apt é realizada pelo comando `apt-get install pacote` (ou `aptitude install pacote`), o apt vai localizar o pacote, realizar o download, instalar o pacote e caso haja a necessidade será aberta uma tela de configuração. Caso haja alguma dependência do pacote o apt vai automaticamente realizar o download (junto com o pacote inicial) e vai instalar estas dependências. Veja abaixo exemplo de instalação:

```
# apt-get install lynx
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  lynx
0 upgraded, 1 newly installed, 0 to remove and 689 not upgraded.
Need to get 1855kB of archives.
After unpacking 4690kB of additional disk space will be used.
Get:1 http://ftp.br.debian.org unstable/main lynx 2.8.5-2 [1855kB]
Fetched 1855kB in 29s (63.7kB/s)
Selecting previously deselected package lynx.
(Reading database ... 145057 files and directories currently installed.)
Unpacking lynx (from ../archives/lynx_2.8.5-2_i386.deb) ...
Setting up lynx (2.8.5-2) ...
```

No exemplo foi instalado o pacote lynx.

8.5.5. Atualizando pacotes Debian via apt

Para atualizar pacotes Debian de sua distribuição utilize o comando `apt-get upgrade`, isto vai atualizar os pacotes instalados para a versão mais atual da distribuição. Geralmente a lista de pacotes é bem grande e existem atualizações diárias.

```
# apt-get upgrade
```

Caso queira atualizar sua distribuição inteira utilize o comando `apt-get dist-`

upgrade:

```
# apt-get dist-upgrade
```

8.5.6. Removendo pacotes Debian via apt

Para remover pacotes Debian de sua distribuição utilize o comando `apt-get remove pacote`, note que se outros pacotes dependerem deste serão também removidos portanto utilize com cuidado, de qualquer forma o apt mostra na tela quais são os pacotes que serão removidos.

```
# apt-get remove lynx
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  lynx
0 upgraded, 0 newly installed, 1 to remove and 690 not upgraded.
Need to get 0B of archives.
After unpacking 4690kB disk space will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 145134 files and directories currently installed.)
Removing lynx ...
```

8.5.7. Outras formas de obter pacotes Debian

Apesar de ser muito útil, o apt não é a única forma de se obter pacotes Debian, eles podem ser obtidos da mesma maneira que pacotes RPM, ou seja executando o download do pacote e instalando manualmente. A boa notícia agora é que os pacotes são facilmente localizados pelo site <http://packages.debian.org> que é o repositório oficial Debian.

8.5.8. Instalando / atualizando pacotes Debian

Vamos instalar o pacote lynx, primeiro encontre este pacote nos repositórios, faça o download. Com o pacote em mãos, instale utilizando a ferramenta `dpkg` com a opção `-i` (de instalar):

```
# dpkg -i lynx_2.8.5-2_i386.deb
Selecting previously deselected package lynx.
(Reading database ... 145057 files and directories currently installed.)
Unpacking lynx (from lynx_2.8.5-2_i386.deb) ...
Setting up lynx (2.8.5-2) ...
```

Pacote instalado.

8.5.9. Resolvendo dependências

Quando encontramos um pacote pelo site <http://packages.debian.org> passamos por sua página de informações, nesta página pode ser encontrada a relação de dependências deste pacote. Veja o exemplo abaixo:

```
# apt-get upgrade
```

File Edit View Go Bookmarks Tools Help

http://pa Go

Slashdot BR-Linux.org VivaoLinux

Other Packages Related to lynx

● = depends
 ◆ = recommends
 ■ = suggests

- [libbz2-1.0](#)
 high-quality block-sorting file compressor library - runtime
- [libc0.3](#) (>= 2.3.2.ds1-4) [hurd-i386]
 GNU C Library: Shared libraries and Timezone data
- [libc6](#) (>= 2.3.2.ds1-4) [not alpha, hurd-i386, ia64]
 GNU C Library: Shared libraries and Timezone data
- [libc6.1](#) (>= 2.3.2.ds1-4) [alpha, ia64]
 GNU C Library: Shared libraries and Timezone data
- [libgnutls11](#) (>= 1.0.16)
 GNU TLS library - runtime library

Done

Estes pacotes devem estar previamente instalados para que o lynx possa funcionar.

Para listar os pacotes instalados digite o comando `dpkg -l` ou utilize o Synaptic / Aptitude.

8.5.10. Forçando a instalação de pacotes Debian

Em algumas ocasiões podemos ter problemas na instalação de pacotes, exatamente como ocorrem com os pacotes RPM, (lembra da história dos pacotes X e Y?), a solução no caso seria forçar a instalação de pacotes. A opção force no caso de pacotes Debian tem mais opções internas, cada situação pode requerer uma solução diferente de force, geralmente utilizando a opção --force-all resolve qualquer parada mas recomendo que leia a opção --force-help (que mostra as outras opções).

8.5.12.Desinstalando pacotes Debian

Para desinstalar pacotes Debian utilize o comando `dpkg -r pacote:`

```
# dpkg -r lynx
(Reading database ... 145134 files and directories currently installed.)
Removing lynx ...
```

Caso algum outro pacote dependa deste, o o gerenciador dpkg não vai permitir a desinstalação, restando duas opções; ou remova pacote por pacote ou force a desinstalação, de qualquer modo recomendo que utilize o apt para estas tarefas.



Exercícios

1. Faça o download do código fonte do programa links, ou copie do diretório / aulas/aula8/, descompacte e instale com a opção --enable-graphics ativada.
2. Caso esteja em uma distribuição com empacotamento RPM, tente instalar o pacote apt-get, habilitando a utilização de apt em outras distribuições.
3. Caso esteja em uma distribuição com empacotamento Debian, instale o pacote alien e tente converter pacotes RPM para Debian.
4. Leia o manual do apt.



Depois de dominarmos o Linux pelo modo textual vamos agora iniciar a utilização do ambiente gráfico. Neste capítulo vamos configurar o ambiente gráfico do zero. Também serão apresentados os window managers, que são os gerenciadores de janelas. São diversas, as mais utilizadas são o GNOME e o KDE e serão explicadas com detalhes, as menos utilizadas serão somente citadas. Este capítulo é essencialmente conceitual e você vai se sentir em casa caso já tenha alguma prática com outro sistema operacional baseado em janelas tal como o Mac OS ou Windows. Ao final deste capítulo o aluno será capaz de:

- Configurar o ambiente gráfico no Linux
- Utilizar o GNOME
- Utilizar o KDE
- Configurar o window manager.



9.1. O que é o servidor X?

O servidor X é o ambiente gráfico do Linux, que gerencia todos os dispositivos que irão interagir graficamente com o usuário, o X controla os seguintes dispositivos de hardware:

- Teclado
- Mouse
- Placa de vídeo
- Monitor

O servidor X permite uma série de opções, como por exemplo a utilização de mais de um monitor para um só desktop, a utilização de ambiente gráfico via rede e também a aceleração 3D para gráficos e jogos. No ano de 2004 o projeto Xfree deixou de ser suportado, o novo projeto que mantém o servidor X é chamado X.org. Existem algumas distribuições que ainda utilizam o Xfree, a configuração dos dois sistemas é semelhante, vamos iniciar a configuração do ambiente gráfico.

9.1.1. Configurando o servidor X

A configuração do X esta contida em um arquivo de texto, os arquivos de texto serão diferentes caso esteja utilizando o Xfree ou o X.org, a localização dos arquivos de configuração também varia de distribuição para distribuição, tornando difícil especificar uma localização genérica, na minha maquina a localização dos arquivos e a seguinte:

| Servidor | Arquivo |
|----------|-------------------------------|
| Xfree | /usr/X11R6/lib/X11/XF86Config |
| X.org | /etc/X11/xorg.conf |

O segredo aqui é localizar os arquivos pelo nome.

Seria muito complicado configurar o ambiente gráfico por um arquivo de texto, imagine configurar a resolução do mouse especificando sua resolução DPI e sua velocidade, para isso existem ferramentas que configuram o servidor X. Estas ferramentas alteram o o arquivo de configuração de seu servidor X, configurando os hardwares especificados no início da seção.

Para o xfree existem duas ferramentas muito utilizadas, a primeira é o `xf86config`, esta ferramenta roda em modo texto e faz uma série de perguntas sobre o tipo de teclado, resolução do monitor, porta do mouse, placa de vídeo e etc. Ao final a ferramenta `xf86config` escreve diretamente no arquivo de configuração. A segunda ferramenta é o `xf86cfg`, apesar de parecidos este último cria uma interface gráfica temporária para configuração e carrega um servidor X genérico com funções que funcionam na maioria dos casos. A configuração neste caso é gráfica, o usuário configura o teclado, o mouse o monitor e a placa de vídeo alterando os valores nos menus e formulários.

Como o Xorg é um servidor gráfico mais novo e atualizado sua configuração é feita graficamente utilizando o comando Xorg com a opção `--configure` (`Xorg --configure`), da mesma forma que o Xfree, abrirá uma janela gráfica com os módulos genéricos que funcionam na maioria dos computadores. Existem ferramentas para a configuração do Xorg como o `xorgcfg`, mas estas não são padrão como no Xfree.

Geralmente a configuração do X tende a ser complicada pois são exigidas muitas informações. Prefira sempre utilizar valores básicos para que seu sistema gráfico funcione, se ele funcionar otimize as opções até que chegue a um ponto desejado.

9.1.2. Iniciando e finalizando o servidor X

Depois de configurado corretamente, o servidor X será iniciado com o comando `startx`, a partir deste momento o servidor X vai ficar hospedado no terminal virtual de número 7, isto quer dizer que ao pressionarmos "Ctrl+Alt+F7" vamos para o terminal gráfico e quando pressionarmos "Ctrl+Alt+FX" (este FX pode variar de F1 à F6) iremos novamente para o console mas sem fechar a parte gráfica.

Quando inicia, o servidor X lê o arquivo `~/.xinitrc` e executa o window manager padrão, veremos adiante o que são Window Managers.

Para reiniciar o servidor X a qualquer hora basta digitar "Ctrl+Alt+Backspace", desta forma o X vai desligar e ligar novamente, em qualquer momento. Quando o X é desligado/reiniciado todos os programas gráficos que rodavam através da parte gráfica também são finalizados, portanto utilize a reinicialização com cuidado. Para finalizar a gui, geralmente o window manager padrão tem um método de desligamento de parte gráfica. Caso precise desligar a parte gráfica manualmente mate o processo chamado Xorg ou Xfree86



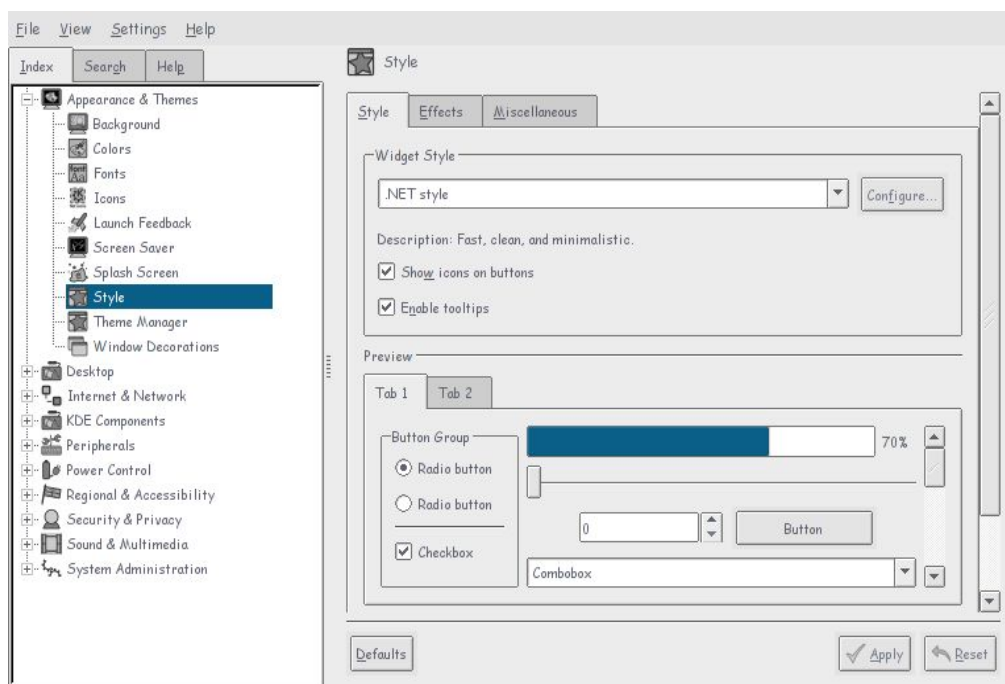
9.2. Window Managers

Apesar do servidor X permite a utilização do modo gráfico, porém isto não basta pois o usuário não pode executar nenhuma ação diretamente no servidor X. Para que o usuário se comunique com o servidor X foi criado um outros softwares que permitem que o usuário se comunique com o servidor X, são os chamados Window Managers, também conhecidos como gerenciadores de janelas. Existem diversos gerenciadores, desde os mais simples com poucos recursos, até os mais sofisticados repletos de funcionalidades e opções. Window Managers que nada mais são do que programas que criam janelas, menus, ícones, área de trabalho entre outros itens de interface gráfica, alguns deles possuem programas específicos. Eles convivem entre si sem problemas, o usuário pode conter mais de um Window Manager em sua máquina caso queira, isto vai ocupar espaço mas ele terá a opção de utilizar um outro Window Manager quando bem entender.



9.3. KDE

O KDE (K Desktop Environment) é o window manager mais utilizado no mundo Linux, tem uma interface agradável, possui muitas ferramentas e também é bem completo no quesito de utilização podendo se equiparar à interface gráfica do sistema operacional Windows. O KDE permite todo tipo de ações, criação de novos arquivos e links, montagem de dispositivos, navegação pela rede, enfim, é um ótimo sistema de janelas para usuários de desktop. Abaixo um screenshot:



KDE 3.3 no Kurumin Linux 4.0

9.3.1. Configurando o KDE

Todas as configurações do KDE podem ser feitas através de seu utilitário de configuração, o `kcontrol`, veja um screenshot:

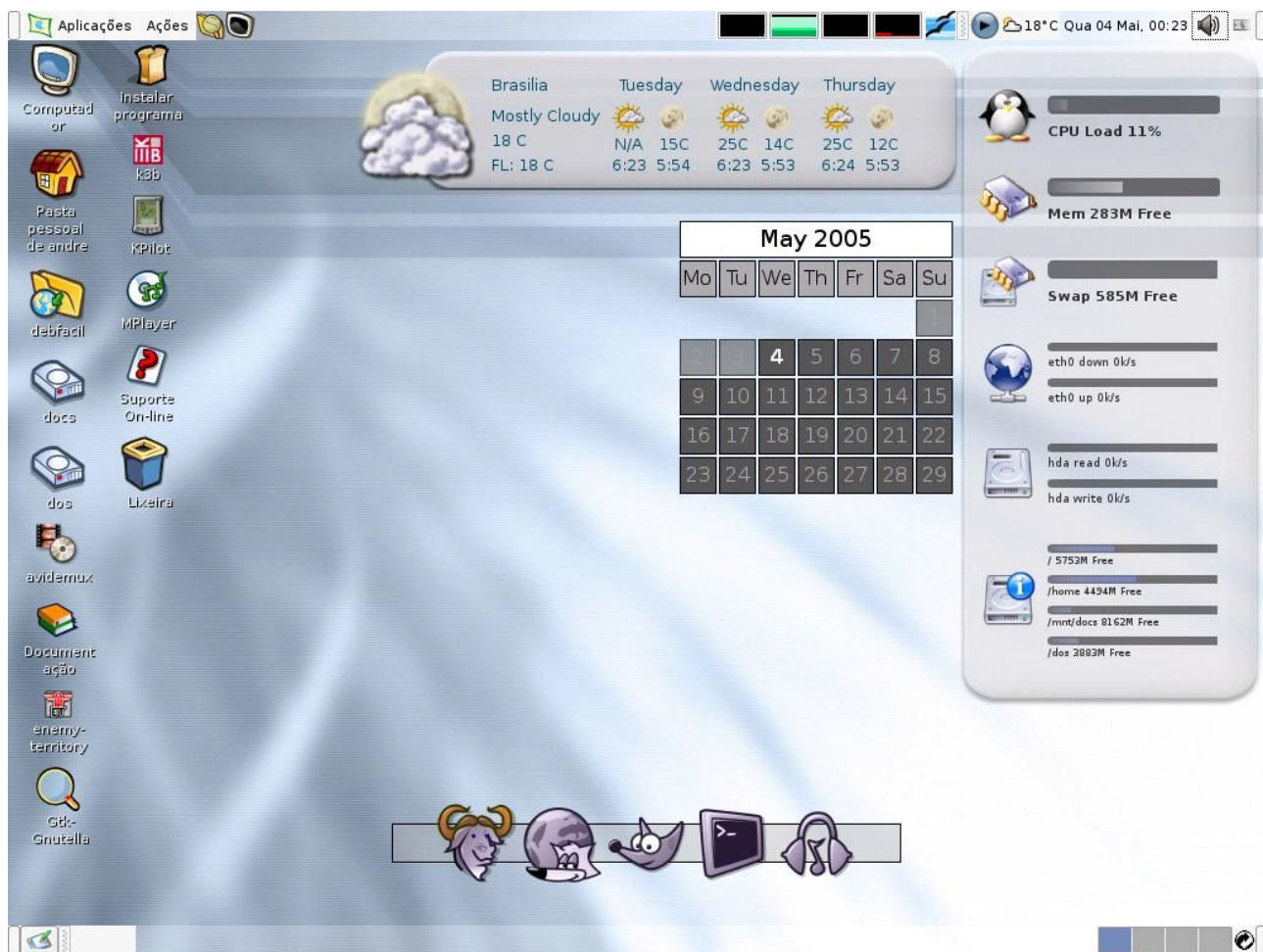
9.3.2. Otimizando o KDE

A interface KDE é bem completa, existe também um site especializado na customização do KDE que fornece novos ícones, novos papéis de parede, estilos e etc. O site é: <http://www.kde-look.org>. A instalação de novos temas e ícones é feita através do Kcontrol, ou através de pacotes rpm ou deb.



9.4. GNOME

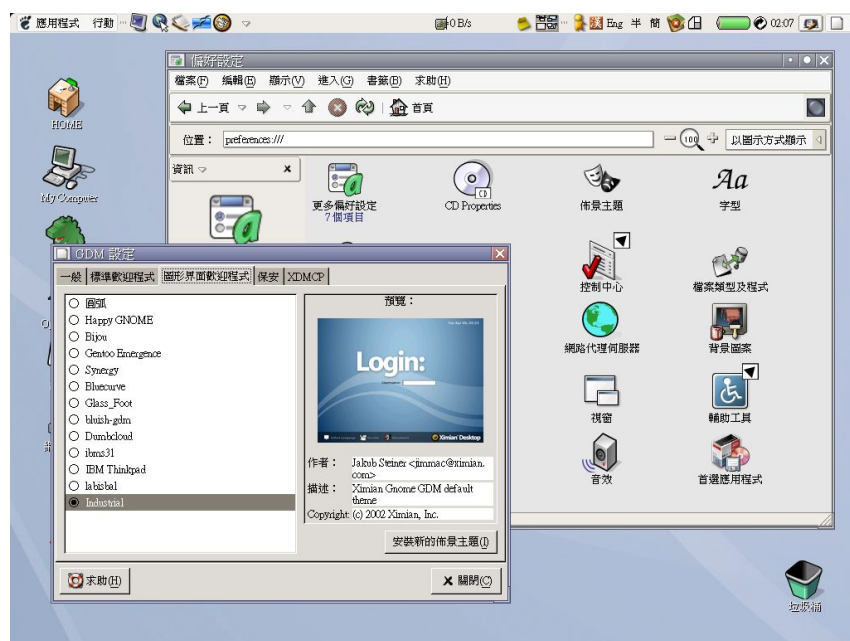
O Gnome também é um Window Manager bem completo, possui uma série de programas, e é muito bonito. Um dos pontos fortes é que o Gnome é ligeiramente mais rápido que o KDE, pois utiliza as bibliotecas GTK para desenho das janelas. É um sistema voltado para o usuário de Desktop. Veja Abaixo um screenshot do Gnome:



Gnome 2.8 no Debian-BR-CDD

9.4.1. Configurando o GNOME

Assim como o KDE, o Gnome possui um utilitário de configuração em sua barra de tarefas:



9.4.2. Otimizando o Gnome

O Gnome também permite a sua otimização, dois ótimos sites são:

<http://art.gnome.org/>

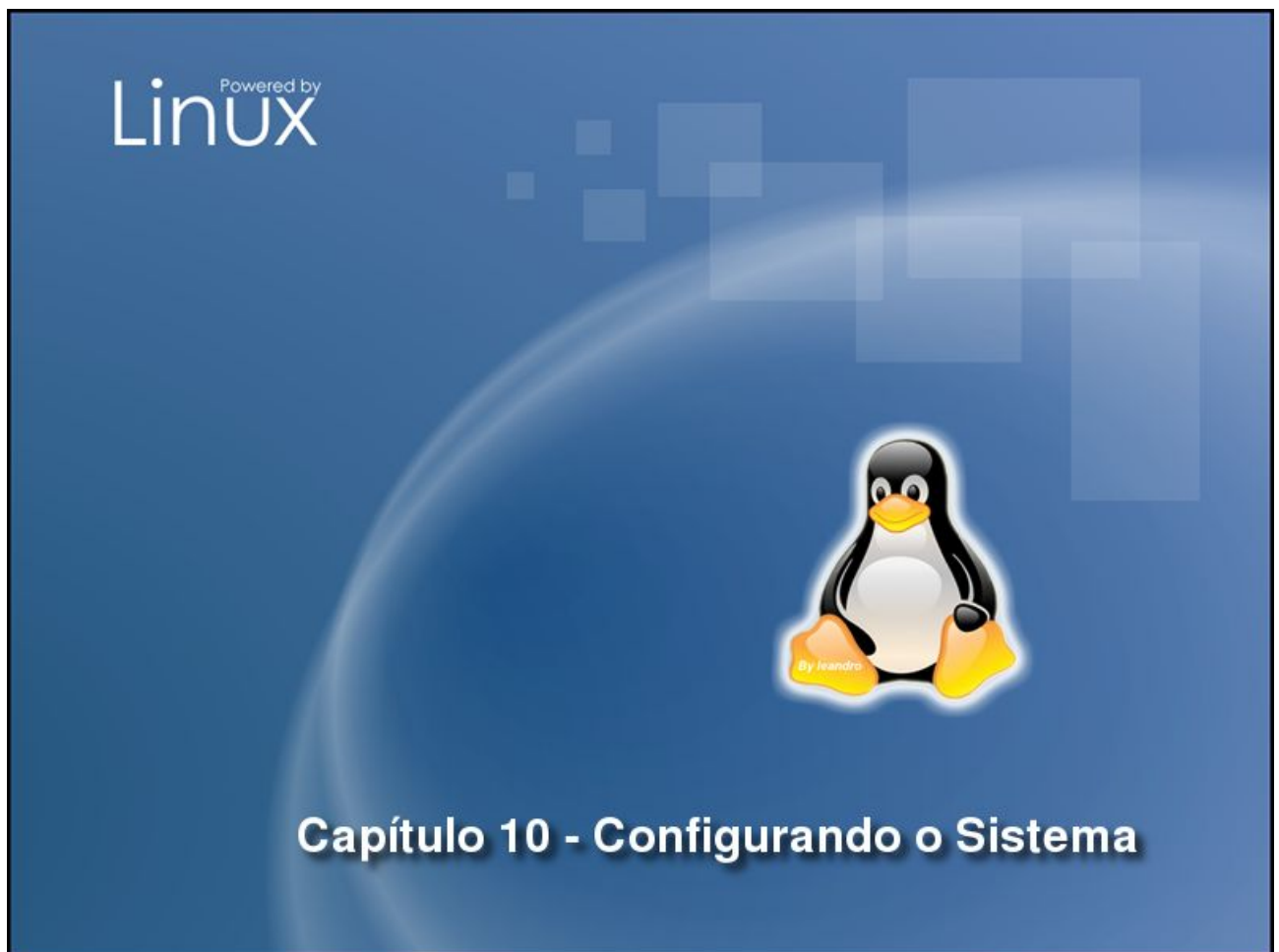
<http://www.gnome-look.org/>



9.5. Outros Window Managers

Existem muitos outros window managers, que agradam os mais diversos gostos. Abaixo seguem os links de suas páginas:

- Fluxbox: <http://www.fluxbox.org/>
- Blackbox: <http://blackboxwm.sourceforge.net/>
- Enlightenment: <http://www.enlightenment.org/>
- Xfce: <http://www.xfce.org/>
- Window Maker: <http://www.windowmaker.org/>
- IceWM: <http://www.icewm.org/>



Hoje em dia, o suporte a hardware no Linux é vasto, é muito raro ver componentes de hardware não funcionar no linux por uma limitação de drivers. A configuração em si não é difícil, e com um pouco de prática torna-se corriqueira. Ao final deste capítulo o aluno será capaz de:

- Configurar placas de rede
- Configurar placas de rede Wi-Fi
- Configurar placas de som
- Configurar modem
- Configurar impressoras
- Configurar scanners
- Configurar multi-funcionais
- Configurar gravadores de CD/DVD
- Configurar câmeras digitais



10.1. Introdução aos módulos do kernel

Antes de mais nada devemos conhecer os módulos do kernel, Um módulo é um componente do kernel que pode ser carregado a qualquer instante pelo usuário, este componente visa uma otimização qualquer ou ativar o suporte a algum dispositivo, inclusive de hardware. Portanto quando queremos instalar uma placa de rede somente colocando a placa no PC não basta para que ela funcione, devem ser carregados os módulos desta placa para que o Kernel possa se comunicar com ela. A ativação de módulos é feita utilizando o comando `modprobe nomedomodulo`. Veja o exemplo abaixo:

```
# modprobe 8139too
```

Para listar os módulos que estão em utilização no momento utilize o comando `lsmod`:

```
# lsmod
Module                Size  Used by
md5                    3840   1
ipv6                  244480  8
pcmcia                 17668   2
irtty_sir              5888   2
sir_dev               13996   1 irtty_sir
...
```

Os módulos ficam guardados no diretório `/lib/modules/(versãodokernel)/` (a versão do kernel pode ser obtida com o comando `uname -r`). Módulos podem ser compilados à parte e instalados, da mesma maneira que instalamos programas normais pelo código fonte. Quando instalamos um novo módulo devemos sempre atualizar a nossa base de módulos utilizando o comando `depmod -a`.

Para incluir a inicialização de um módulo no boot altere o arquivo `/etc/modules` adicionando o nome do módulo, recomendo que leia o manual de `modules.conf`.



10.2. Configurando placas de rede

A configuração de interfaces de rede é tranqüila, basta carregar o módulo da placa para que ela comece a funcionar. Caso não saiba qual módulo carregar ou não saiba o modelo de sua placa utilize o comando `lspci`, este utilitário vai fazer uma busca em `/proc/` (que é o sistema virtual criado pelo kernel) e vai retornar todos os componentes detectados pela motherboard, inclusive os que não foram configurados. Neste output procure pela palavra Ethernet, veja o exemplo abaixo:

```
d3v1l@Shinji:~/info/idepac$ lspci
0000:00:00.0 Host bridge: Silicon Integrated Systems [SiS] 650/M650 Host (rev 80)
0000:00:01.0 PCI bridge: Silicon Integrated Systems [SiS] Virtual PCI-to-PCI bridge (AGP)
0000:00:02.0 ISA bridge: Silicon Integrated Systems [SiS] SiS962 [MuTIOL Media IO] (rev 14)
0000:00:02.1 SMBus: Silicon Integrated Systems [SiS]: Unknown device 0016
0000:00:02.3 FireWire (IEEE 1394): Silicon Integrated Systems [SiS] FireWire Controller
0000:00:02.5 IDE interface: Silicon Integrated Systems [SiS] 5513 [IDE]
0000:00:02.6 Modem: Silicon Integrated Systems [SiS] AC'97 Modem Controller (rev a0)
0000:00:02.7 Multimedia audio controller: Silicon Integrated Systems [SiS] Sound Controller (rev a0)
0000:00:03.0 USB Controller: Silicon Integrated Systems [SiS] USB 1.0 Controller (rev 0f)
0000:00:03.1 USB Controller: Silicon Integrated Systems [SiS] USB 1.0 Controller (rev 0f)
0000:00:03.2 USB Controller: Silicon Integrated Systems [SiS] USB 1.0 Controller (rev 0f)
0000:00:03.3 USB Controller: Silicon Integrated Systems [SiS] USB 2.0 Controller
0000:00:04.0 Ethernet controller: Silicon Integrated Systems [SiS] SiS900 PCI Fast Ethernet (rev 90)
0000:00:0a.0 CardBus bridge: Texas Instruments PCI1410 PC card Cardbus Controller (rev 02)
0000:01:00.0 VGA compatible controller: Silicon Integrated Systems [SiS] 65x/M650/740 PCI/AGP VG
A Display Adapter
```

Como podemos verificar, a placa ethernet é uma Sis 900, vamos procurar no Google/Linux (<http://www.google.com.br/linux>) esta placa, com poucos minutos podem ser encontradas diversas referências. Agora basta carregar o módulo:

```
# modprobe sis900
```

Logo após carregar o módulo, utilize o comando dmesg e leia as últimas linhas para ver se sua placa foi carregada com sucesso:

```
# dmesg
...
...
...
sis900.c: v1.08.07 11/02/2003
ACPI: PCI interrupt 0000:00:04.0[A] -> GSI 7 (level, low) -> IRQ 7
divert: allocating divert_blk for eth1
eth0: Realtek RTL8201 PHY transceiver found at address 1.
eth0: Using transceiver found at address 1 as default
eth0: SiS 900 PCI Fast Ethernet at 0xa000, IRQ 7, 00:11:2f:32:6a:77.
```

A interface agora vai se chamar eth0, isso indica que a placa foi instalada e configurada.



10.3. Configurando placas de rede Wi-Fi

A configuração de redes wi-fi é semelhante, porém, algumas destas placas no entanto não possuem módulos no kernel oficial. O que resta então é compilar o módulo diretamente do código fonte. Uma ótima fonte de referência na instalação de novas placas de rede é o tutorial de instalação de placas wi-fi no Linux, do Guia do Hardware: <http://www.guiadohardware.net/tutoriais/092/> está bem completo e cobre diversas placas diferentes.

Algumas placas não tem suporte oficial ao Linux ainda, a melhor maneira de fazê-las funcionar é utilizando o ndiswrapper, este programa emula um driver do windows, transformando-o em módulo, o kernel vai pensar que está trabalhando com um módulo nativo e a placa vai funcionar, o desempenho pode cair um pouco e algumas funcionalidades da placa também podem ficar de fora mas a placa vai funcionar basicamente. Existem mais informações sobre o Ndiswrapper no site mencionado acima.



10.4. Configurando placas de som

As placas de som no Linux são instaladas através do projeto ALSA (Advanced Linux Sound Architecture). São um conjunto de drivers e bibliotecas que permitem a utilização da maioria das placas de som disponíveis hoje no mercado. O site oficial do projeto ALSA é <http://www.alsa-project.org/>, lá podem ser encontrados tutoriais de instalação das placas e relação de placas suportadas, vamos instalar uma como exemplo:

Com o comando `lspci` procure a sua placa de som:

```
# lspci
...
0000:00:02.7 Multimedia audio controller: Silicon Integrated Systems
[SiS] Sound Controller (rev a0)
...
```

Ok, vimos que é uma Sis, agora devemos baixar os drivers, bibliotecas e utilitários, no site do projeto ALSA, faça o download da última versão dos arquivos `alsa-driver`, `alsa-lib`, `alsa-utils`. Devemos também verificar o site com informações sobre a instalação de sua placa.

Crie o diretório `/usr/src/alsa` e mande os para este diretório.

```
# mkdir /usr/src/alsa
# mv alsa* /usr/src/alsa
```

Descompacte o `alsa-driver`:

```
/usr/src/alsa# tar -xvjf alsa-driver-1.0.9rc4a.tar.bz2
...
...
```

No diretório do driver, comece o processo de instalação:

```
/usr/src/alsa/alsa-driver-1.0.9rc4a# ./configure
/usr/src/alsa/alsa-driver-1.0.9rc4a# make
/usr/src/alsa/alsa-driver-1.0.9rc4a# make install
```

Ok, placas instaladas, agora ajuste a permissão para os dispositivos de audio para que todos possam ler e escrever nesta placa:

```
# chmod a+rw /dev/dsp /dev/mixer /dev/sequencer /dev/midi
```

O repita o processo com alsa-lib:

```
/usr/src/alsa/# tar -xvjf alsa-lib-1.0.9rc4.tar.bz2
/usr/src/alsa/# cd alsa-lib-1.0.9rc4
/usr/src/alsa/alsa-lib-1.0.9rc4# ./configure
/usr/src/alsa/alsa-lib-1.0.9rc4# make
/usr/src/alsa/alsa-lib-1.0.9rc4# make install
```

E também com alsa-utils:

```
/usr/src/alsa/# tar -xvjf alsa-utils-1.0.9rc4.tar.bz2
/usr/src/alsa/# cd alsa-utils-1.0.9rc4
/usr/src/alsa/alsa-utils-1.0.9rc4# ./configure
/usr/src/alsa/alsa-utils-1.0.9rc4# make
/usr/src/alsa/alsa-utils-1.0.9rc4# make install
```

Finalmente carregue os módulos da sua placa:

```
# modprobe snd-xxxx
```

O xxxx é o nome da placa, procure no site do alsa, e cheque o `dmesg` para ver se sua placa foi corretamente carregada.



10.5. Configurando modem

No mercado existem dois tipos de modems, os modems ISA antigos e os novos Winmodems (também chamados de softmodems). O Linux tem amplo suporte aos Modems convencionais ISA, sua instalação é fácil e não são necessários módulos para que este seja configurado, os modems são detectados como portas Seriais (ttySX).

Já a configuração de winmodems é muito complicada, estes modems não funcionam por hardware (não são hardware de verdade) funcionam por software, e seu software foi desenvolvido para Windows, muitos dos softmodems foram mais com alguns destes modems possuem software para linux mas a cada mudança de kernel suas configurações deixam de funcionar. Para que um Winmodem funcione no Linux é necessário antes de tudo muita paciência e também informação, por serem muitos casos diferentes não será coberta na apostila a instalação de Winmodems, verifique no site <http://www.linmodems.org> mais informações sobre este tipo de modem.

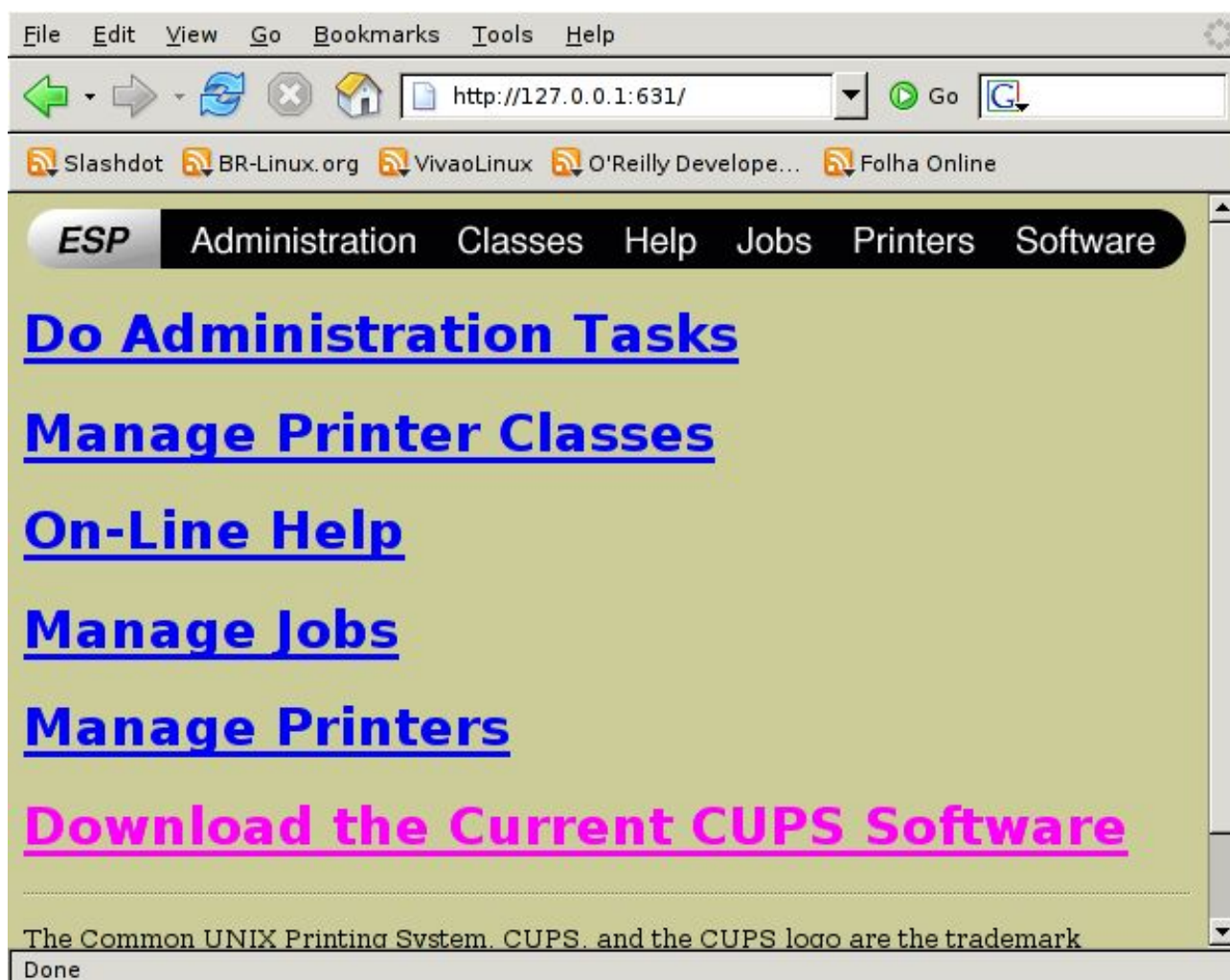


10.6. Configurando impressoras

A configuração de impressoras no linux também é bem tranquila, é feita através do CUPS, que é um utilitário de impressão do Unix. A maioria das distribuições já vem com o CUPS e os drivers de impressão, mas caso não estejam instalados pegue os pacotes binários do cups ou então compile do código fonte, o site oficial é <http://www.cups.org>. Depois que o cups estiver instalado inicie o serviço:

```
# /etc/init.d/cups start
```

Agora para a instalação de uma nova impressora, abra o navegador e digite o endereço <http://127.0.0.1:631> :



1. A instalação é feita via web, para adicionar uma impressora clique em "Do administration Tasks" insira como username: root e como senha: a senha de root.
2. Na interface de administração clique em "Add Printer".
3. Agora no campo "Name" digite o nome da impressora sem espaços por exemplo Laserjet1015. No campo "Location" não é obrigatório mas caso queria especificar o nome do computador ou departamento onde está a impressora o local é aqui, não deixa de ser uma boa prática de administração. O campo "Description" também não é obrigatório mas pode ser inserida uma descrição sobre a impressora. Acabando aqui clique em "Continue".
4. Nesta página, selecione onde se encontra o dispositivo fisicamente. Se for na porta paralela (como é mais comum) selecione "Parallel Port #1", Para uma impressora USB selecione a porta desejada. Existe também a opção de adicionar uma impressora que está na rede, caso seja este o caso na próxima página digite o endereço da impressora. Vamos supor que nossa impressora seja paralela, então selecionaremos "Parallel Port #1". Acabando aqui clique em "Continue".
5. Nesta página devemos selecionar o fabricante da impressora, vamos supor que nossa impressora seja uma HP. Clique em "Continue".
6. Selecione agora o modelo de sua impressora, vamos supor que nossa impressora seja uma LaserJet 1015, procure na lista e selecione esta impressora. Algumas impressoras possuem dois tipos de driver, o hpijs e o ljet4 por exemplo. Dê sempre

preferência para o hpijs pois este possui o driver mais estável. Acabando aqui clique em "Continue". Ok a impressora foi adicionada, agora clique em "Printers" na barra superior e mande imprimir uma página de teste clicando em "Print Test Page".

Uma informação importante: para que impressoras USB funcionem corretamente é necessário que o módulo usbprinter esteja carregado.



10.7. Configurando scanners

A configuração de scanners também é feita através do SANE (Scanner Access Now Easy), como no caso do CUPS, a maioria das distribuições atuais possuem o SANE, se já não tiver instalado procure os pacotes binários nos repositórios de sua distribuição ou então compile o código fonte. O site oficial do SANE é <http://www.sane-project.org/>. Um excelente manual de configuração de scanners pode ser encontrado em : <http://howtos.linux.com/howtos/Scanner-HOWTO/index.shtml>.

Caso o scanner seja USB (que são a maioria absoluta no mercado), basta iniciar o sane que o seu scanner será automaticamente detectado. Para detectar utilize o comando `sane-find-scanner`.

```
$ sane-find-scanner
found USB scanner (vendor=0x03f0, product=0x2f11) at libusb:002:002
```



10.8. Configurando multi-funcionais

A configuração de multi-funcionais é feita exatamente da mesma forma, que equipamentos únicos. A diferença é que precisamos antes instalar uma biblioteca que comunique ao kernel que o periférico possui mais de uma funcionalidade. O nome desta biblioteca é HPOJ (Hewlett Packard Office Jet) que apesar de ser desenvolvido e suportado pela HP, possui suporte a outros multi-funcionais. Para instalar este módulo procure os pacotes binários de sua distribuição.

Depois de instalado utilize o comando `ptal-init setup`, este utilitário vai perguntar se é um dispositivo paralelo, USB ou network (rede), vamos supor que seja um dispositivo USB veja o resultado:

```
# ptal-init setup
Probe for USB-connected devices ([y]/n)? y
Probing "/dev/usb/lp0"...
Found "psc 1200 series"
with serial number "BR43B4G0XX5H".
This device is already set up as "mlc:usb:psc_1200_series".
```

Ok foi detectado, agora desligue e ligue o equipamento (multifuncional) e finalmente digite `dmesg` para verificar se foi detectado.



10.9. Configurando gravadores de CD/DVD

A configuração de gravadores de cds e dvds varia de acordo com a versão utilizada do Linux. A maioria das distribuições atuais vem com o utilitário cdrecord e o cdrdao. Ambos são necessários para a instalação e gravação de cds, novamente ressaltado, caso não esteja instalada baixe os pacotes.

Para o kernel 2.4 é necessário que se carregue o módulo ide-scsi:

```
# modprobe ide-scsi
```

Outra coisa necessária é que se passe um parâmetro no boot do sistema, caso esteja usando LILO adicione a seguinte linha ao arquivo /etc/lilo.conf:

append="hdx=ide-scsi"

O trecho do arquivo ficará desta forma:

```
image=/vmlinuz
    label=Linux
    read-only
    append="hdx=ide-scsi"
```

Caso esteja utilizando o GRUB, deverá ser adicionado o parâmetro diretamente na linha do kernel, o trecho do arquivo que será alterado ficará desta forma:

```
title Fedora Core (2.6.5-1.358smp)
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.5-1.358smp ro root=LABEL=/ hdx=ide-scsi
    initrd /boot/initrd-2.6.5-1.358smp.imgptal-init setup
```

Uma coisa, eu utilizei o drive como hdx em ambos os exemplos, não será este o drive, o x no caso significa a letra da unidade, para verificar qual é a sua unidade de disco utilize o comando `dmesg | grep '^hd.:'`, veja o exemplo abaixo:

```
# dmesg | grep '^hd.:'
hda: IC25N040ATMR04-0, ATA DISK drive
hdb: ASUS SCB-2408, ATAPI CD/DVD-ROM drive
hda: max request size: 1024KiB
hda: 78140160 sectors (40007 MB) w/1740KiB Cache, CHS=16383/255/63, UDMA
(100)
hda: cache flushes supported
hdb: ATAPI 24X DVD-ROM CD-R/RW drive, 2048kB Cache, UDMA(33)
```

No exemplo foi verificado que o drive hdb é o gravador de Cds. Agora reinicie o computador, e utilize o `cdrecord` para detectar seu drive:

```
# cdrecord -scanbus
Cdrecord 1.10 (i686-pc-linux-gnu) Copyright (C) 1995-2001 Jörg Schilling
Linux sg driver version: 3.1.20
Using libscg version 'schily-0.5'
scsibus0:
 0,0,0      0) 'IDE-CD  ' 'R/RW 12x8x32  ' ' 3.0' Removable CD-ROM
 0,1,0      1) *
 0,2,0      2) *
 0,3,0      3) *
 0,4,0      4) *
 0,5,0      5) *
 0,6,0      6) *
 0,7,0      7) *
```

Anote os três números que aparecem na primeira coluna ao lado do drive, no caso do exemplo o número é 0,0,0. Pronto o Cdr no kernel 2.4 foi configurado.

No kernel 2.6 a coisa muda, o suporte já vem habilitado no kernel, para verificar seu gravador utilize o comando `cdrecord -scanbus`, veja o exemplo abaixo:

```
# cdrdao scanbus
Cdrdao version 1.1.9 - (C) Andreas Mueller <andreas@daneb.de>
SCSI interface library - (C) Joerg Schilling
Paranoia DAE library - (C) Monty
Check http://cdrdao.sourceforge.net/drives.html#dt for current driver
tables.
Using libscg version 'schily-0.8'
ATA:0,1,0      ASUS      , SCB-2408      , 1.2B
```



10.10. Configurando câmeras digitais

Câmeras digitais no Linux é semelhante à configuração das memórias flash usb, basta carregar o módulo `usb-storage`:

```
# modprobe usb-storage
```

Agora o dispositivo será reconhecido na interface SCSI e poderá ser montado normalmente, utilize o `dmesg` para verificar a unidade, provavelmente será a **/dev/sda**.

Nas versões mais novas do kernel o módulo `usb-storage` foi substituído por um driver nativo, exatamente como os CDRs. Para acessar a unidade basta montar a unidade **/dev/ubx** (sendo esse x a unidade correta), novamente invoque o `dmesg` para saber a mesma.



É muito importante que os computadores se comuniquem entre-si formando redes, a própria internet é uma grande rede e neste capítulo vamos aprender a configurar o acesso à redes no Linux. Ao final deste capítulo o você será capaz de:

- Configurar uma rede
- Acessar a internet via modem
- Acessar a internet via banda-larga
- Acessar uma rede windows
- Acessar uma rede Linux



11.1. Configurando a rede

A configuração de uma rede TCP-IP é feita através da ferramenta `ifconfig`, nela especificamos de uma só vez todos os parâmetros para que possamos subir a rede. a formatação deste arquivo segue a seguinte regra:

```
ifconfig interface <numero.do.ip.aqui> netmask <mascara.de.sub.rede> up
```

Vamos subir a interface de rede:

```
# ifconfig eth0 192.168.0.30 netmask 255.255.255.0 up
```

Isto sobre a rede e temos acesso agora a outros hosts.



11.2. Configurando via DHCP

A configuração via DHCP é mais fácil, supondo que existe algum servidor DHCP na rede o endereçamento passa a ser automático, basta utilizar o comando `dhclient` (o programa `dhclient` deve estar instalado):

```
# dhclient eth0
Internet Software Consortium DHCP Client 2.0p15
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.

Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html

Listening on LPF/eth0/00:00:00:00:00:00
Sending on   LPF/eth0/00:00:00:00:00:00
Sending on   Socket/fallback/fallback-net
DHCPREQUEST on eth1 to 255.255.255.255 port 67
DHCPACK from 192.168.0.50
SIOCADDRT: File exists
bound to 192.168.0.60 -- renewal in 21600 seconds.
```



11.3. Acessando a Internet

Para acessar a internet devemos ter um provedor (teoricamente), devemos configurar também o nosso DNS, que resolve os números IPs transformando-os em nomes. Por exemplo `uol.com.br` é o nome do IP `200.221.2.45`, experimente digitar este IP no navegador e veja o que acontece. Adicione a seguinte linha ao arquivo `/etc/resolv.conf`:

```
nameserver 200.204.0.10
```

11.3.1. Discando via modem

Para discar e conectar à internet via modem você deve criar um link simbólico da porta onde está o modem até o arquivo **/dev/modem**, veja o exemplo abaixo:

```
# ln -s /dev/ttyS1 /dev/modem
```

Aps a criação do atalho, configure o programa **wvdial** (o discador padrão do Linux) utilizando o programa **wvdialconf**, veja o exemplo abaixo:

```
# wvdialconf /etc/wvdial.conf  
...
```

A saída desse comando é bem grande portanto não será incluída nesta apostila. No arquivo recém criado (**/etc/wvdial.conf**) altere as configurações para as oferecidas pelo seu provedor, veja o exemplo abaixo:

```
[Dialer Defaults]  
Phone = 5552233  
Username = bruno  
Password = bruno  
New PPPD = yes  
Modem = /dev/modem  
Baud = 115200  
Init1 = AT&F1&C1&D2X3  
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0  
ISDN = 0  
Modem Type = Analog Modem  
Dial Prefix = 0
```

O arquivo acima é só um exemplo, altere somente as opções de username, senha e telefone. Com tudo configurado inicie o **wvdial** depois de discado, verifique se a conexão está ok com o comando **ifconfig**:

```
# ifconfig  
ppp0      Link encap:Point-to-Point Protocol  
          inet addr:123.234.123.1  P-t-P:123.0.0.1  Mask:255.255.255.255  
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1  
          ...
```

11.3.2. Acessando via banda-larga

A internet banda-larga é acessada diretamente pela interface de rede, basta configurar uma rede como vimos no início do capítulo, no caso de uma rede com IP fixo devemos passar as informações que o provedor nos passou, a única diferença que pode acontecer aqui é que o provedor peça um gateway. A configuração do gateway consiste em utilizar o comando **route add default gw <numero.ip.do.gateway>** veja o exemplo abaixo:

```
# route add default gw 192.168.0.254
```

Caso sua conta no provedor seja de IP dinâmico devemos utilizar o DHCP exatamente como no início do capítulo, tudo é configurado automaticamente.

Alguns serviços de banda larga utilizam um método de autenticação chamado de PPPoE (PPP over Ethernet), o que significa que você deve entrar com usuário e senha antes de acessar. Para se conectar desta forma você deverá instalar os pacotes pppoe, pppoeconf e ppp, (distribuições com pacotes rpm possuem o rp-pppoe que entra no lugar do pppoe e pppoeconf). A configuração de distribuições baseadas em Debian consiste na ferramenta pppoeconf (que faz tudo automaticamente), já distribuições baseadas em Red Hat utilizam o comando adsl-setup. A configuração dos dois é bem simples qualquer dúvida pode ser acompanhada pelo manual.



11.4. Acessando uma rede Windows

Máquinas com sistemas Windows e Linux conversam entre si via rede e tem total interoperabilidade, servidores Linux podem ter clientes Windows trabalhando, podemos compartilhar impressoras, diretórios, arquivos entre os dois sistemas. Grande parte das empresas que utilizam Linux atualmente possuem redes mistas com estações Windows e Linux, nessa seção vamos aprender basicamente como compartilhar arquivos e diretórios com uma rede Windows básica.

11.4.1. Apresentando o Samba

O Projeto Samba consiste em um conjunto de ferramentas que tornam a comunicação de uma máquina Linux com uma máquina Windows possível. Isto é um grande negócio pois como vimos no capítulo 1, o Linux tem suas vantagens como servidor. É um projeto robusto e completo, possui vasta documentação e tem diversas opções de trabalho. Para mais informações visite o site <http://www.samba.org>.

As distribuições incluem em seus repositórios pacotes binários com o samba, caso tenha que instalar manualmente os pacotes são samba, samba-common, samba-doc.

11.4.2. Configurando o Samba via SWAT

Existe uma ferramenta gráfica para a configuração do samba, geralmente as distribuições não habilitam este pacote por default, mas ele pode ser instalado sem problemas através de pacotes.

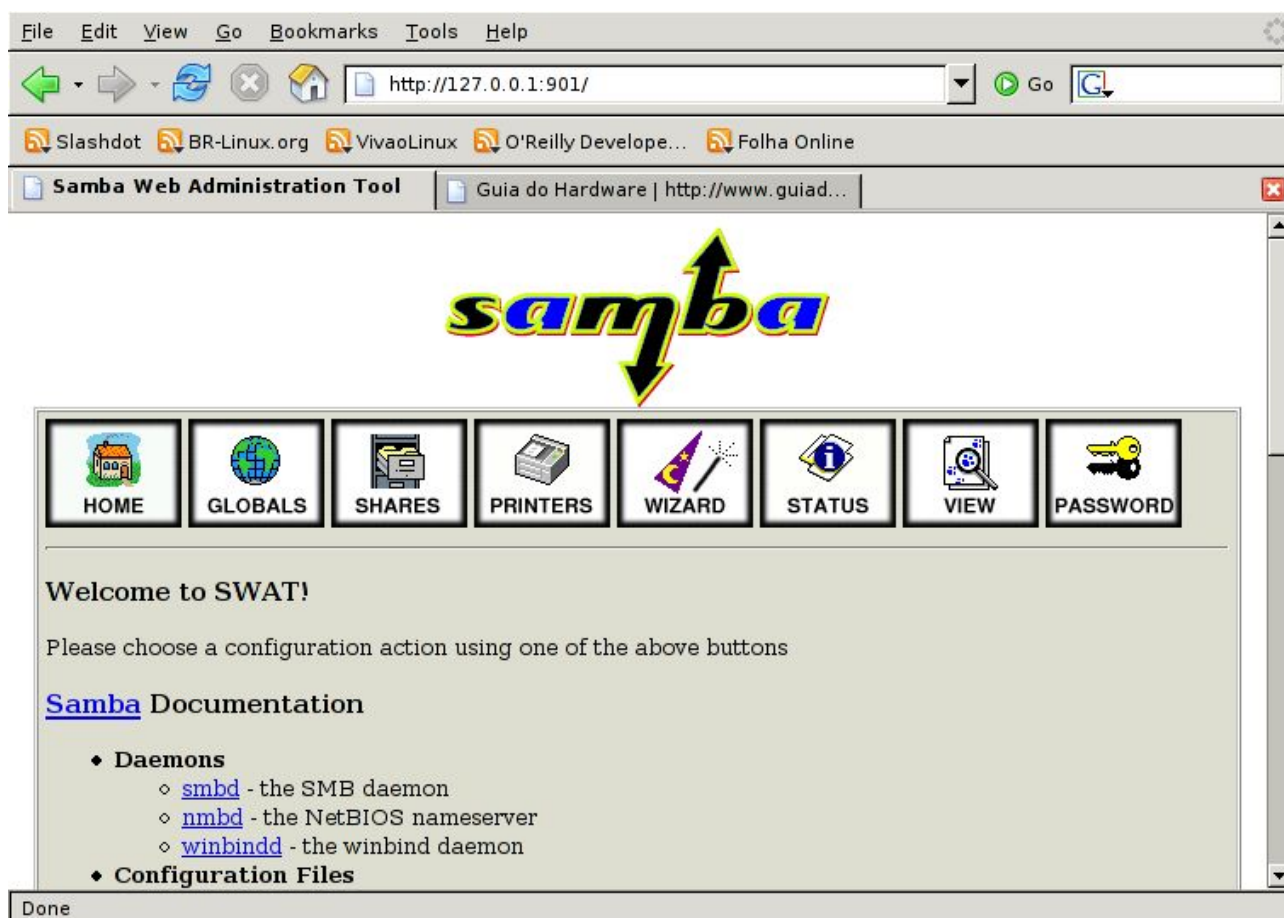
Depois de instalado precisamos configurar o acesso (temporário pois é meio inseguro), no arquivo /etc/inetd procure por uma linha que começa com swat. caso encontre descomente esta linha, caso não encontre adicione a linha abaixo:

```
swat stream tcp nowait.400 root /usr/sbin/tcpd /usr/sbin/swat
```

Reinicie o servidor inetd:

```
# /etc/init.d/inetd restart
```

Feito isto acesse o endereço <http://127.0.0.1:901> em seu navegador:



Interface do SWAT

Através dele podemos administrar o samba. As configurações principais estão na seção global, veja abaixo uma configuração básica de exemplo:

Seção Global:

| | |
|-------------------|--|
| workgroup | Grupo de trabalho da rede |
| netbios name | Nome da máquina na rede |
| security | Sistema de segurança da rede, selecione USER |
| encrypt passwords | A partir do Windows 98 as senhas eram Encriptadas, selecione Yes |
| passwd backend | Gerenciador de senhas smbpasswd |
| username map | arquivo com as senhas dos usuários /etc/samba/smbusers |
| os level | Prioridade entre estações na rede selecione 23 |
| preferred master | Auto |
| local master | Auto |
| domain master | Auto |

Configuração básica realizada, lembre-se esta configuração é preferencial quando o Linux for cliente na rede.

11.4.3. Montando uma unidade de rede

Montar uma unidade de rede Windows é semelhante a montar qualquer outra unidade de disco, o que difere aqui é o sistema de arquivos e as opções utilizadas. Vamos supor que o computador TOAD roda windows e tem o diretório c:\ compartilhado, para montar este diretório:

```
# mount -t smbfs -o username=usuario //numero.ip.do.toad/C /mnt/toad
Password:
```

Digite a senha do usuário e pronto, a estação windows pode ser acessada agora. O usuário que mencionamos acima é o usuário cadastrado no windows. Para adicionar esta entrada ao /etc/fstab, insira a linha abaixo (substituindo os valores) no final do arquivo:

```
//numero.ip.do.toad/C      /mnt/toad      smbfs      username=usuario,noauto 0 0
```

Agora podemos acessar a unidade simplesmente digitando `mount /mnt/toad`.

11.4.4. Compartilhando diretórios

Voltando no SWAT para compartilhar um diretório clique na seção "Shares", especifique um nome para o compartilhamento e clique em "Create Share"

Veja abaixo as principais opções da seção Share:

| | |
|------------|---|
| comment | Comentário do compartilhamento. |
| path | Caminho do diretório por exemplo /alunos/aulaX |
| read only | Permissão para leitura e escrita |
| browseable | Deixa o compartilhamento como visível ou invisível (se for invisível o compartilhamento ainda vai existir mas não vai aparecer) |

Quando acabar clique em "Commit Changes" para que as alterações façam efeito. Os usuários que acessarem o compartilhamento devem estar cadastrados no sistema também e adicionados ao samba, para isso na seção "Password" preencha os campos adicionando o usuário como mostra a figura abaixo:



Clique em "Add New User" e pronto o usuário pode acessar o sistema.

11.4.5. Compartilhando impressoras

Na seção "Printers" escolha uma das impressoras na caixa de seleção, caso queira selecionar todas de uma vez selecione "printers" e clique em "Choose Printer". Abaixo seguem as principais opções da seção Printers:

| | |
|------------|---|
| comment | Comentário da impressora. |
| path | Caminho do spool de impressão, mantenha /var/spool/samba . |
| printable | Permissão de impressão. |
| browseable | Deixa a impressora como visível ou invisível (se for invisível a impressora ainda vai existir mas não vai aparecer) |



11.5. Acessando uma rede Linux

O serviço de redes padrão do Linux é o NFS (Network File System), que permite a montagem de blocos de dispositivos de rede, é um sistema de redes muito rápido e seguro. Para a utilização do NFS precisamos do pacote nfs-common e nfs-kernel-server. Quando utilizamos uma rede Linux-Linux com compartilhamento NFS nenhuma configuração é necessária para acessarmos os arquivos remotos, o único requisito é que as duas máquinas tenham o NFS instalado.

11.5.1. Montando uma unidade de rede

Montamos compartilhamentos NFS da mesma forma que qualquer unidade de disco, vamos supor que o computador remoto YOSHI possua o diretório **/var/www/** compartilhado via NFS, para acessarmos utilizamos a seguinte sintaxe:

```
# mount -t nfs -o soft numero.ip.do.yoshi:/var/www/ /mnt/yoshi
```

Foi montado com sucesso, para adicionar esta entrada ao /etc/fstab, insira a linha abaixo (substituindo os valores) no final do arquivo:

```
numero.ip.do.yoshi:/var/www /mnt/yoshi nfs soft 0 0
```

Agora podemos acessar a unidade simplesmente digitando `mount /mnt/yoshi`. Montei com a opção **soft** pois quando um host cai e esta opção não está especificada o cliente trava até o host voltar à ativa.

11.5.2. Compartilhando uma unidade de rede

Para compartilhar uma unidade de rede via NFS devemos alterar o arquivo /etc/exports, a sintaxe é a seguinte:

```
<diretorio> <numero.ip.da.rede/mascara.de.sub.rede>(opcoes)
```

Por exemplo o host Yoshi, vamos ver como ele compartilha os arquivos:

```
/var/www/ 192.168.0.0/255.255.255.0(sync,rw)
```

Leia o manual exports para saber o que são e para que server as opções sync, rw.

Agora para exportar (compartilhar) os diretórios, utilize o comando `exportfs -a`:

```
# exportfs -a
```

Para ver os arquivos compartilhados utilize o comando `exports` sem opções.



Este apêndice tem não faz parte do curso, mas visa orientar o aluno na instalação do sistema operacional GNU/Linux, tendo como base a instalação da distribuição Fedora Core 2. Serão abordadas nesse capítulo as principais rotinas de instalação em um guia prático seguindo passo-a-passo uma instalação comum.

- Instalar qualquer distribuição do sistema GNU/Linux.
- Iniciar uma instalação do zero ou atualizar uma instalação já realizada.
- Fazer um Sistema GNU/Linux conviver com outro Sistema Operacional num mesmo disco rígido.
- Fazer Dual Boot (escolher na inicialização qual sistema operacional vai iniciar).
- Definir quais pacotes e programas serão instalados.



A.1. Considerações Iniciais

A instalação da maioria das distribuições Linux é simples e descomplicada. Como já foi visto antes, iremos utilizar a distribuição Fedora, porém, a instalação de todas as distribuições é semelhante, todas tem de passar pelos mesmos processos básicos (seleção de layout de teclado, seleção de idioma, particionamento do disco rígido, seleção de pacotes, configuração da conta de root, instalação do boot loader e etc), portanto, quem sabe instalar uma distro aprende com facilidade instalar qualquer outra.

Algumas máquinas antigas funcionam melhor com distribuições mais simples, sem muitos recursos, portanto antes de instalar uma distribuição cheque se a máquina vai rodar a mesma satisfatoriamente.

Por ser um sistema completo, algumas distribuições colocam muitos programas que vêm junto com o Linux, isso nos proporciona um sistema completo pois tudo o que precisamos está incluso em nossa distribuição. Por outro lado, essa inclusão de programas torna o sistema extremamente grande, as distros atuais tem diversos Cds. É também necessário escolher quais programas serão instalados no sistema para não ficarmos com um computador cheio de inutilidades. Em um Linux básico instalado de uma forma correta conseguimos gastar até 200 MB.

A.1.1. Onde conseguir o Linux?

Como já foi dito antes, o Linux é software livre e conseguimos baixá-lo da internet de graça, a pergunta é como?

Existem diversos sites que proporcionam o download de ISOs de Linux, uma ISO é uma imagem de um CD gravada em um arquivo. Tudo depende da distribuição que você procura, uma boa fonte de pesquisa é o site: <http://www.linuxiso.org> nele você encontra listas de sites de download divididos por distribuição. Nos sites das distribuições você encontra downloads também, divididos por países. É necessário ter gravador de CD caso queira baixar e instalar em Cds.

Se sua conexão é discada, fica inviável baixar 3 Cds de instalação, então eu recomendo que os compre no site: <http://www.guiadohardware.net/cd/linux/gnu.php> o preço dos Cds é relativamente baixo 10 Reais e você recebe o Linux em casa. Note bem que o Linux continua sendo gratuito, é cobrada unicamente a mão de obra e as mídias.

A.1.2 Outros meios de instalação.

Existem outros meios de instalação sem a utilização de Cds, estes meios porém são pouco utilizados devido à dificuldades ou inviabilidades. Estes meios são por instalação via rede (denominada "Instalação via NFS"), a qual já temos um servidor Linux e este compartilha os arquivos de instalação via rede, e via FTP, a qual instalamos diretamente da internet as distros acessando os servidores onde estão hospedadas.

Estas situações devem ser utilizadas quando for conveniente a instalação mas implica em uma dificuldade maior. Existem também outros meios específicos de algumas distribuições mas foge do intuito do curso explicar estes meios, caso queira mais informações utilize ferramentas de busca na internet.



A.2 Bootando o Linux

Boot significa, a grosso modo, iniciar a máquina. Para que possamos instalar o Linux precisamos fazer com que a máquina inicie o instalador contido no CD, este é o processo de bootar o Linux. Todas as máquinas mais novas podem iniciar o CD automaticamente, o que torna o processo de boot automático, basta colocar o CD e iniciar a máquina e pronto, o instalador entra em ação.

Já em algumas máquinas mais antigas ou máquinas que não possuem drive de CD-Rom a instalação dificulta já que não conseguimos iniciar o instalador automaticamente. Neste caso caímos nas situações descritas no tópico anterior (2.1), caso a máquina destino seja antiga recomendo que instale distribuições mais simples, como a Slackware por exemplo, informações de como instalar via disquete estão em <http://www.slackware.com>, e se sua máquina for nova mas não tenha drive de CD caímos na situação 2.1.2, uma boa fonte de informações de como construir um disquete de boot e instalar via rede (o Fedora) está nesta página:

<http://www.redhat.com/archives/fedora-docs-list/2004-August/msg00031.html>



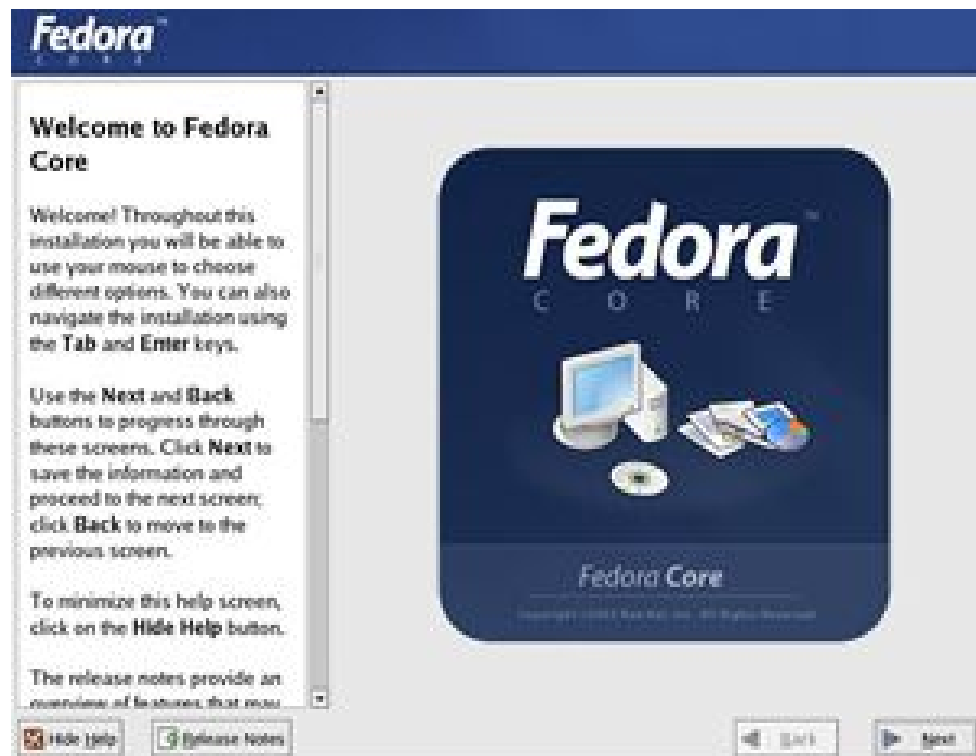
A.3 O Instalador

Com o CD-Rom carregado no sistema o instalador será iniciado. Uma tela semelhante a esta deverá aparecer:



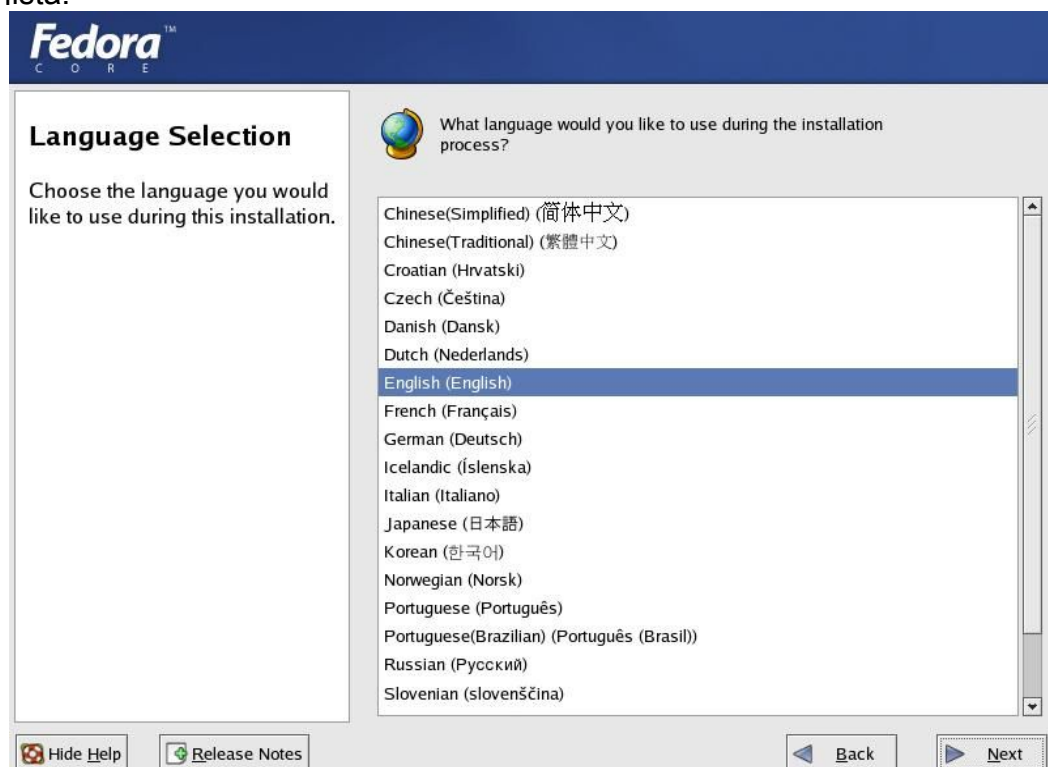
Basta digitar `Enter` que o Kernel será carregado e o processo de instalação terá início. Só a nível de curiosidade o instalador do Fedora Linux (e do Red Hat também) se chama Anaconda.

Quando o anaconda iniciar nos será a apresentada a tela de boas vindas:

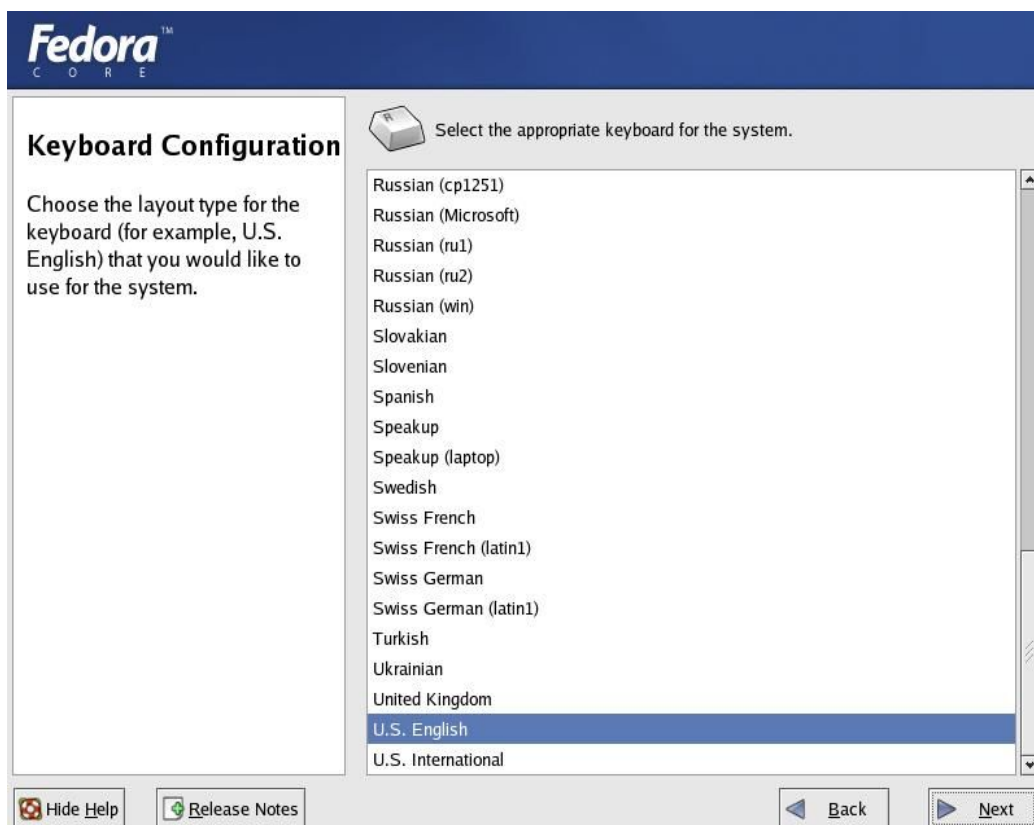


esta tela é unicamente informativa, no canto esquerdo temos notas dos autores da distribuição.

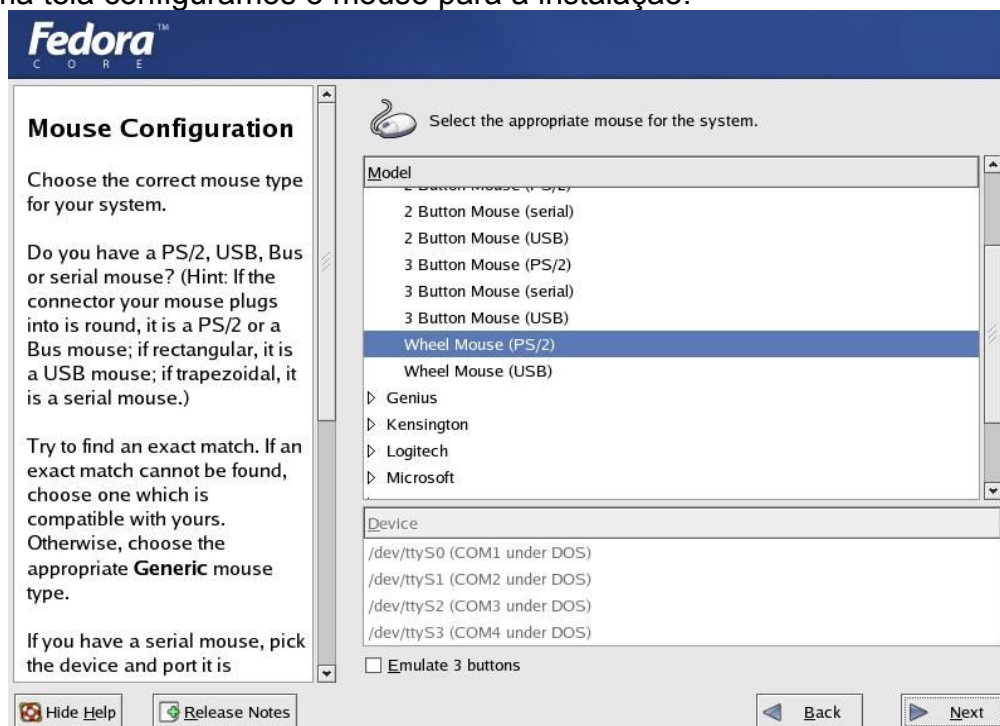
Na próxima tela escolhemos a linguagem da instalação. Note que o português do Brasil está na lista.



Na próxima tela podemos selecionar o layout do teclado que utilizaremos:



Na próxima tela configuramos o mouse para a instalação:



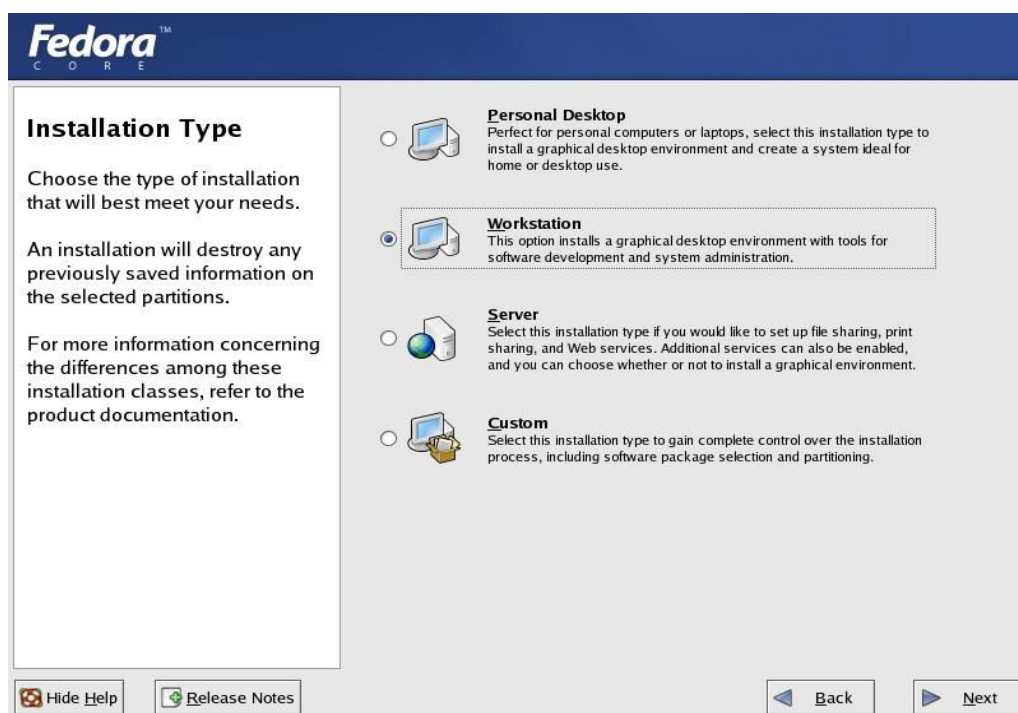
Na próxima tela será perguntado se você deseja atualizar uma instalação antiga ou se deseja simplesmente iniciar uma instalação nova.



Até agora não foi necessária muita atenção na instalação. O passo seguinte é importante pois vai definir o perfil de instalação do sistema.

A.3.1 Definindo o perfil da instalação

A instalação de um sistema GNU/Linux varia dependendo da finalidade do sistema e do tipo de máquina. Por exemplo, existem programas que não são necessários de serem instalados em uma Workstation (Estação de trabalho de uma rede), e que devem ser instalados em um Server (Servidor de serviços). Um servidor não precisa de ambiente gráfico já uma Workstation precisa (na maioria dos casos), o instalador do Fedora resolve esta questão lhe oferecendo alguns perfis de escolha:



Personal Desktop: Computador pessoal de casa.

Workstation: Estação de trabalho de uma rede corporativa.

Server: Servidor de serviços, pode ser um servidor da rede ou de internet.

Custom: Todo e qualquer outro PC que não se enquadra acima. Aqui você seleciona os pacotes manualmente.

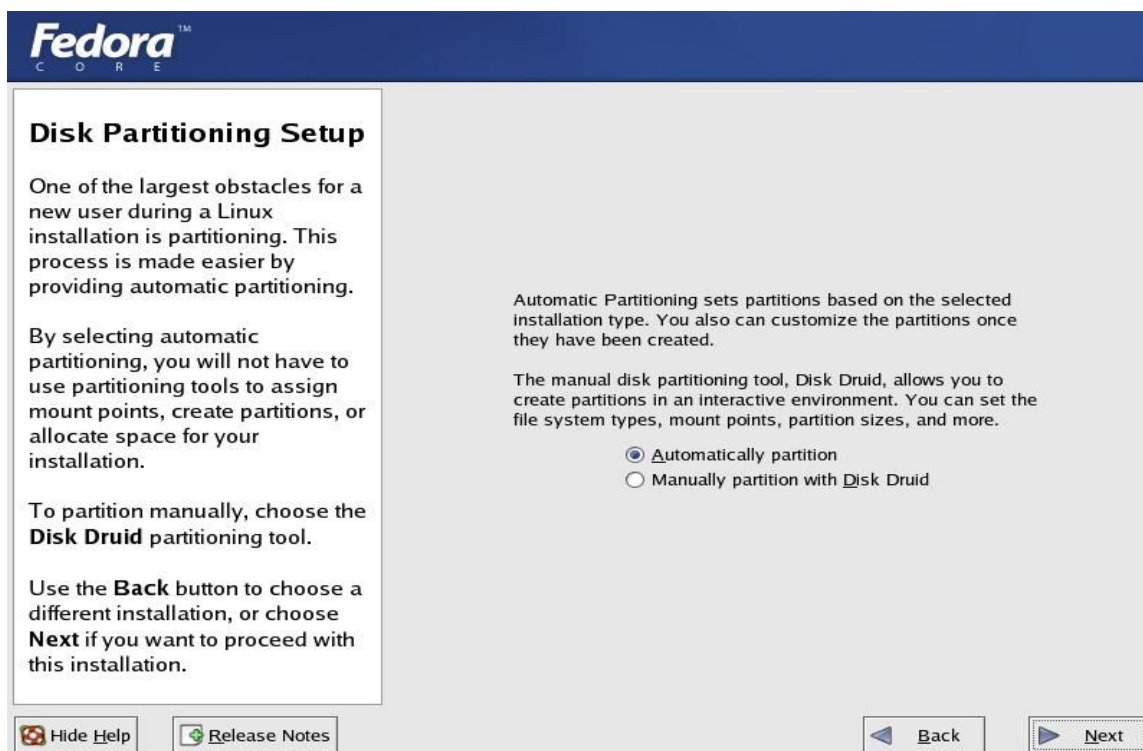
O depois de selecionado o instalador vai automaticamente selecionar os pacotes indicados, no entanto, você ainda pode indicar os programas que queira em uma etapa mais adiante.

Eu recomendo fortemente que você selecione um dos perfis acima citados pois terá poucos problemas de configuração, analise criticamente a finalidade do computador que você está instalando.

A.3.2 Particionando o disco rígido

Aqui vem uma parte que geralmente assusta quem é iniciante, particionando o disco. Como foi dito anteriormente, precisamente no Capítulo 1 tópico 1.2 (Características), o sistema de arquivos do Linux é o Ext3, portando o disco precisa ser formatado neste sistema de arquivos para que possamos instalar e operar o linux adequadamente, existem outros sistemas de arquivos suportados pelo linux como o Journaling, Ext2, ReiserFS e etc. porém eu recomendo o Ext3.

Dessa forma, precisamos dividir, especificar e formatar a unidade de disco rígido em que vamos instalar o Linux. A ferramenta de gerenciamento de partições do instalador se chama Diskdruid:



- Caso queira que o instalador particione seu disco automaticamente selecione a primeira opção.
- Caso queira customizar as partições no seu disco rígido selecione a segunda opção, o particionamento manual não será coberto por este curso justamente por ser de nível intermediário a avançado.

Na próxima tela, o instalador pergunta como deve fazer o particionamento automático:

FedoraTM
C O R E

Automatic Partitioning

Automatic partitioning allows you to have some control concerning what data is removed (if any) from your system.

To remove only Linux partitions (partitions created from a previous Linux installation), select **Remove all Linux partitions on this system**.

To remove all partitions on your hard drive(s) (this includes partitions created by other operating systems such as Windows 95/98/NT/2000), select **Remove all partitions on this system**.

Before automatic partitioning can be set up by the installation program, you must choose how to use the space on your hard drives.

I want to have automatic partitioning:

- ☒ Remove all Linux partitions on this system
- ☐ Remove all partitions on this system
- ☐ Keep all partitions and use existing free space

Select the drive(s) to use for this installation:

| | | | |
|-------------------------------------|-----|----------|-----------|
| <input checked="" type="checkbox"/> | hda | 76293 MB | ST380011A |
|-------------------------------------|-----|----------|-----------|

☐ Review (and modify if needed) the partitions created

Hide Help Release Notes Back Next

- Na 1ª opção ele pergunta se deve remover somente as partições Linux do sistema (caso já tenha partições Ext3 formatadas no disco)
- Na 2ª opção ele pergunta se deve apagar todo o conteúdo contido no disco. Use essa opção com muito cuidado pois vai deletar TUDO no disco.
- Na 3ª opção ele pergunta se deve manter todas as partições e usar o espaço livre.

Se for utilizar um outro sistema operacional que já esteja instalado na máquina utilize as opções 1 ou 3. Isto varia de situação, por exemplo se você já tinha algum Linux instalado na sua máquina então utilize a opção 1, agora se não tinha e tem espaço livre no disco (espaço não formatado) utilize a opção 3. **Cuidado!:** Ter espaço livre aqui significa estritamente espaço não formatado, ou seja espaço que nenhum outro sistema operacional possa acessar.

Se for utilizar somente o Linux utilize a opção 2 já que não há a necessidade de preservar nenhum dado.

O próximo passo cobre a instalação de um bootloader, é ele quem vai gerenciar a escolha de que sistema operacional você escolhe toda vez que liga o micro.

A.3.3 Configurando o Bootloader (GRUB)

O bootloader é um gerenciador de boot, quando você liga a máquina ele te mostra um menu com os sistemas operacionais e lhe pergunta qual deles será iniciado.

O bootloader padrão do Fedora é o GRUB que é bem completo e bonito, na tela

Boot Loader Configuration

By default, the GRUB boot loader will be installed on the system. If you do not want to install GRUB as your boot loader, select **Change boot loader**.

You can also choose which OS (if you have more than one) should boot by default. Select **Default** beside the preferred boot partition to choose your default bootable OS. You will not be able to move forward in the installation unless you choose a default boot image.

You may add, edit, and delete the boot loader entries by selecting a partition with your

The GRUB boot loader will be installed on /dev/hda. [Change boot loader](#)

You can configure the boot loader to boot other operating systems. It will allow you to select an operating system to boot from the list. To add additional operating systems, which are not automatically detected, click 'Add.' To change the operating system booted by default, select 'Default' by the desired operating system.

| Default | Label | Device |
|-------------------------------------|-------------|-----------|
| <input type="checkbox"/> | DOS | /dev/hda2 |
| <input checked="" type="checkbox"/> | Fedora Core | /dev/hda5 |

[Add](#)
[Edit](#)
[Delete](#)

A boot loader password prevents users from changing options passed to the kernel. For greater system security, it is recommended that you set a password.

☐ [Use a boot loader password](#) [Change password](#)

☐ [Configure advanced boot loader options](#)

[Hide Help](#) [Release Notes](#) [Back](#) [Next](#)

abaixo é mostrado o menu de configuração do GRUB:

Ele vai detectar automaticamente os sistemas operacionais da máquina. Caso queira editar algum nome ou parâmetro clique em [Edit](#).

A.3.4 Configurando a Rede e Firewall.

Na próxima tela se o instalador detectou corretamente a sua placa de rede, ele irá perguntar se deseja configurar a rede. Caso saiba todos os parâmetros de sua rede edite à vontade. Para fins didáticos editaremos isso manualmente depois.

Fedora
C O R E

Network Configuration

Any network devices you have on the system will be automatically detected by the installation program and shown in the **Network Devices** list.

To configure the network device, first select the device and then click **Edit**. In the **Edit Interface** screen, you can choose to have the IP and Netmask information configured by DHCP or you can enter it manually. You can also choose to make the device active at boot time.

If you do not have DHCP client access or are unsure as to what this information is, please

Network Devices

| Active on Boot | Device | IP/Netmask |
|-------------------------------------|--------|------------|
| <input checked="" type="checkbox"/> | eth0 | DHCP |

Edit

Hostname

Set the hostname:

☒ automatically via DHCP

☐ manually (ex. "host.domain.com")

Miscellaneous Settings

Gateway:

Primary DNS:

Secondary DNS:

Tertiary DNS:

Hide Help **Release Notes** **Back** **Next**

Imediatamente depois o instalador irá lhe perguntar se deseja ativar o firewall clique na segunda opção, (Enable firewall, Habilitar Firewall) e deixe como está. Este curso não abordará maiores detalhes sobre firewall por ser um tema de nível avançado.

A.3.5 Idioma Padrão / Zona de Horário.

Nas próximas telas o instalador lhe pergunta respectivamente qual idioma será instalado no computador e em que cidade se localiza o mesmo.

Fedora
C O R E

Additional Language Support

Select a language to use as the default language. The default language will be the language used on the system once installation is complete. If you choose to install other languages, it is possible to change the default language after the installation.

The installation program can install and support several languages. To use more than one language on your system, choose specific languages to be installed, or select all languages to have all available languages installed on the system.

Select the default language for the system: English (USA)

Select additional languages to install on the system:

- ☐ English (Denmark)
- ☐ English (Great Britain)
- ☐ English (Hong Kong)
- ☐ English (India)
- ☐ English (Ireland)
- ☐ English (New Zealand)
- ☐ English (Philippines)
- ☐ English (Singapore)
- ☐ English (South Africa)
- ☒ English (USA)
- ☐ English (Zimbabwe)
- ☐ Estonian
- ☐ Faroese (Faroe Islands)
- ☐ Finnish
- ☐ French (Belgium)
- ☐ French (Canada)
- ☐ French (France)
- ☐ French (Luxemburg)
- ☐ French (Switzerland)

Select All **Select Default Only** **Reset**

Hide Help **Release Notes** **Back** **Next**

Fedora
C O R E

Time Zone Selection

Set your time zone by selecting your computer's physical location.

On the interactive map, click on a specific city (marked by a yellow dot) and a red X will appear indicating your selection. You can also scroll through the available list and choose a time zone.

You can also scroll through the city list and choose your desired time zone.

You can also select the **System Clock uses UTC** option. (UTC, also known as GMT, will allow your system to properly handle daylight-

Please select the nearest city in your timezone:

America/Los_Angeles - Pacific Time

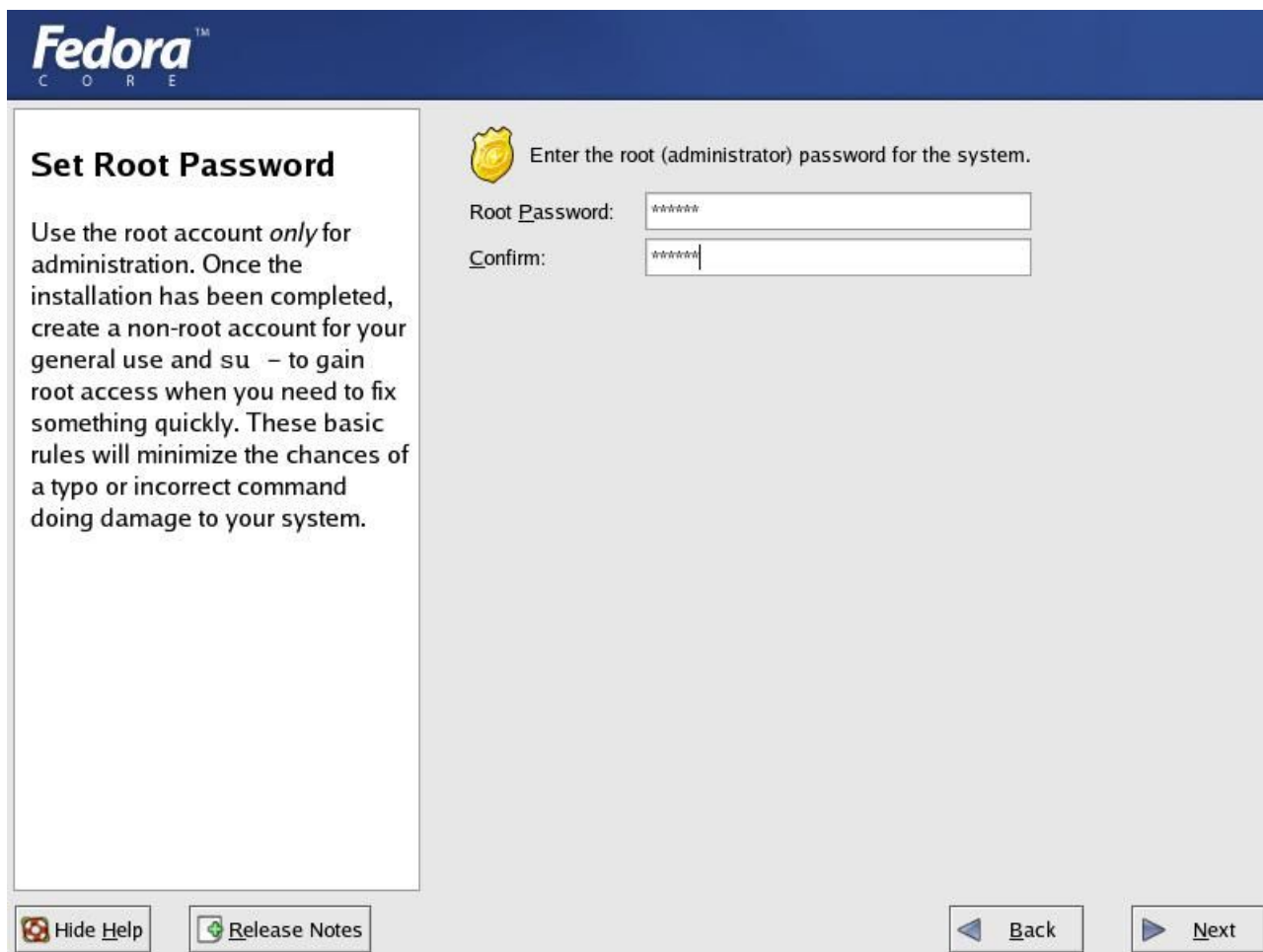
| Location | Description |
|---------------------|---|
| America/Lima | Pacific Time |
| America/Los_Angeles | Pacific Time |
| America/Louisville | Eastern Time - Kentucky - Louisville area |

☐ System clock uses UTC

Hide Help **Release Notes** **Back** **Next**

A.3.6 Senha de root

O usuário root será explicado mais adiante, porém já é requerida uma senha. Digite uma senha e não a divulgue a ninguém, com esta senha o acesso ao sistema é ilimitado. Lembre-se de digitar uma senha que não esqueça.



The screenshot shows the 'Set Root Password' screen in the Fedora Core installer. On the left, a text box explains the importance of the root password and advises creating a non-root user. On the right, there are two input fields for the root password, labeled 'Root Password:' and 'Confirm:'. Both fields contain six asterisks. At the bottom, there are buttons for 'Hide Help', 'Release Notes', 'Back', and 'Next'.

Fedora™
C O R E

Set Root Password

Use the root account *only* for administration. Once the installation has been completed, create a non-root account for your general use and `su -` to gain root access when you need to fix something quickly. These basic rules will minimize the chances of a typo or incorrect command doing damage to your system.

Enter the root (administrator) password for the system.

Root Password:

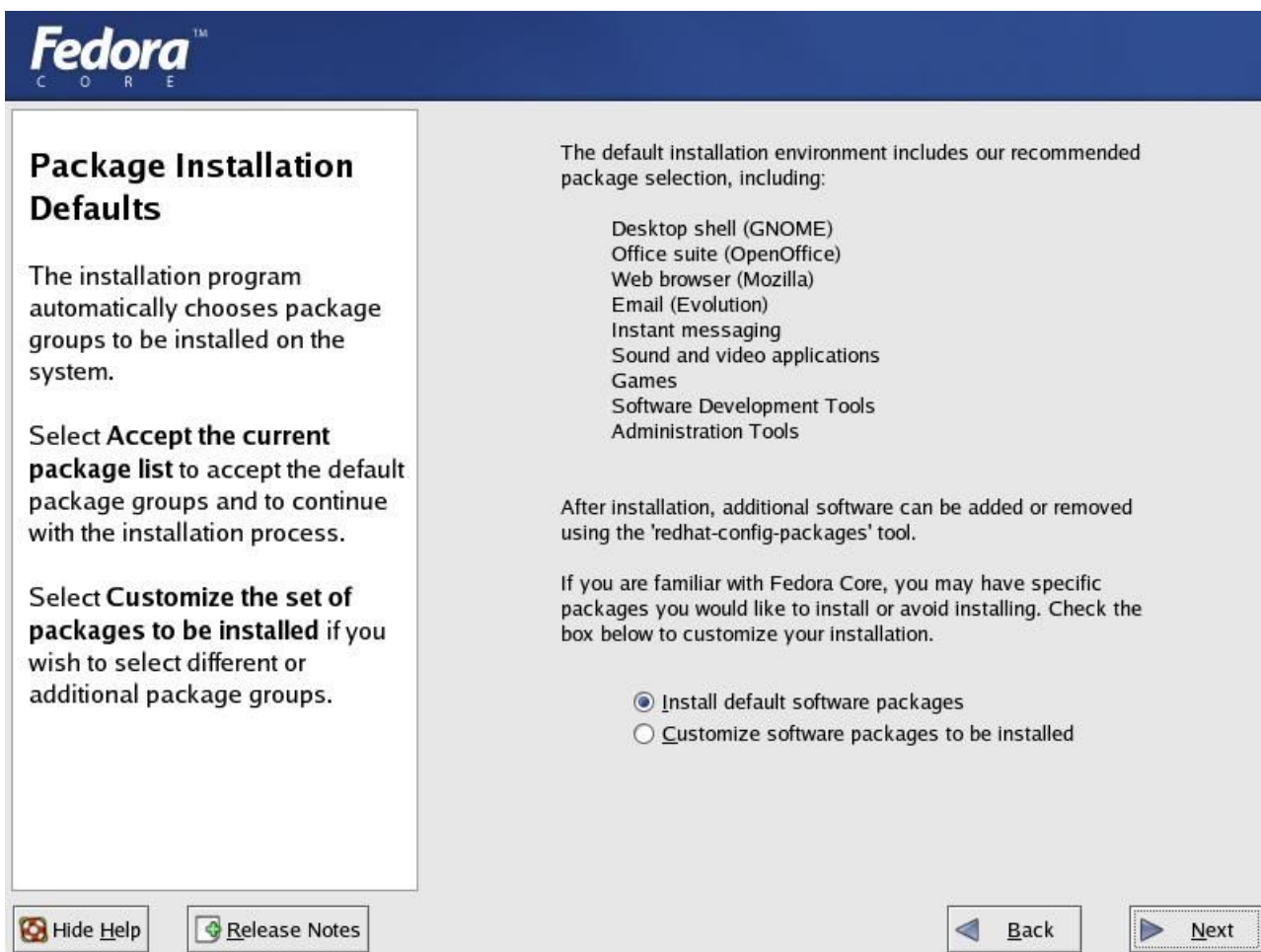
Confirm:

Hide Help Release Notes Back Next

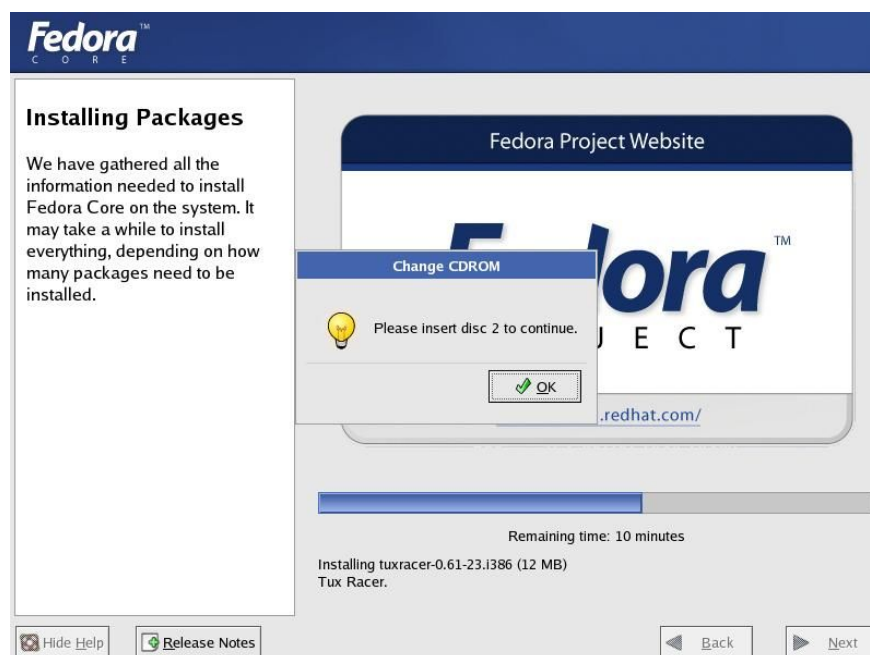
A.3.7 Finalizando

Com todas as configurações já realizadas e a senha de root definida, o instalador lhe pergunta se deseja adicionar mais algum pacote (lembra que isso foi dito no item 2.3.1? Bom aqui será realizada esta customização, caso seja necessária ou por gosto do usuário, vale a pena dar uma olhada.

Caso queira instalar imediatamente clique na primeira opção, se quiser explorar os pacotes e programas e eventualmente instalar mais algum clique na segunda.



Na próxima tela o sistema lhe informa que foi tudo concluído e ele irá instalar o sistema, cuidado pois esta é sua última chance antes que o instalador inicie a cópia de arquivos.



A cópia de arquivos será iniciada, e pode demorar um pouco dependendo da capacidade da máquina. No meio da cópia, o instalador vai lhe pedir para inserir o próximo CD de instalação, retire o anterior, coloque o próximo e prossiga normalmente. Ao final da instalação o instalador vai lhe apresentar a tela de conclusão e vai pedir para reiniciar a máquina, retire o CD que estiver na bandeja e reinicie.

